(2)

NAVAL POSTGRADUATE SCHOOL

Monterey, California

UNITED STATES NAVY
NAVAL POSTGRADUATE SCHOOL

THESIS

SECURING APPLICATIONS
IN PERSONAL COMPUTERS:
THE RELAY RACE APPROACH

by

James Michael Wright

September, 1991

Thesis Advisor: Moshe Zviran

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE			
1a REPORT SECURITY CLASSIFICATION UNCLASSIFIED		1b RESTRICTIVE MARKINGS	
2a. SECURITY CLASSIFICATION AUTHORITY		3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.	
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE			
4. PERFORMING ORGANIZATION REPORT NUMBER(S)		5 MONITORING ORGANIZATION REPORT NUMBER(S)	
6a. NAME OF PERFORMING ORGANIZATION Naval Postgraduate School	6b OFFICE SYMBOL *(If applicable)* 55	7a. NAME OF MONITORING ORGANIZATION Naval Postgraduate School	
6c. ADDRESS *(City, State, and ZIP Code)* Monterey, CA 93943-5000		7b. ADDRESS *(City, State, and ZIP Code)* Monterey, CA 93943-5000	
8a NAME OF FUNDING/SPONSORING ORGANIZATION	8b OFFICE SYMBOL *(If applicable)*	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER	
8c. ADDRESS *(City, State, and ZIP Code)*		10 SOURCE OF FUNDING NUMBERS	

Program Element No	Project No	Task No	Work Unit Accession Number

11 TITLE *(Include Security Classification)*
SECURING APPLICATIONS IN PERSONAL COMPUTERS: THE RELAY RACE APPROACH

12 PERSONAL AUTHOR(S) Wright, James M.

13a. TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT *(year, month, day)*	15 PAGE COUNT
Master's Thesis	From To	September 1991	107

16 SUPPLEMENTARY NOTATION
The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

17 COSATI CODES			18 SUBJECT TERMS *(continue on reverse if necessary and identify by block number)*
FIELD	GROUP	SUBGROUP	Security, Personal Computer, Applicaion Level, Microcomputer.

19 ABSTRACT *(continue on reverse if necessary and identify by block number)*

This Thesis reviews the increasing need for security in a personal computer (PC) environment and proposes a new approach for securing PC applications at the application layer. The Relay Race Approach extends two standard approaches: data encryption and password access control at the main program level, to the subprogram level by the use of a special parameter, the "Baton." The applicability of this approach is demonstrated in an original Basic application and an existing DbaseIV application, representing both third generation language (3GL) and fourth generation language (4GL) environments. The Approach can add to overall network security in the PC LAN environment as well The Approach is successful and proposed enhancements can strengthen the Approach.

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT		21 ABSTRACT SECURITY CLASSIFICATION
[✓ UNCLASSIFIED/UNLIMITED] ☐ SAME AS REPORT ☐ DTIC USERS		Unclassified
22a NAME OF RESPONSIBLE INDIVIDUAL Moshe Zviran	22b TELEPHONE *(Include Area code)* 408-646 2489	22c OFFICE SYMBOL AS/ZV

DD FORM 1473, 84 MAR

83 APR edition may be used until exhausted
All other editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE
Unclassified

Securing Applications in Personal Computers: The Relay Race Approach

by

James Michael Wright
Lieutenant Commander, United States Navy
B.S., University of Florida, 1980

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SYSTEMS MANAGEMENT

from the

NAVAL POSTGRADUATE SCHOOL
September 1991

Author: [illegible]
James Michael Wright

Approved by: [illegible]
Moshe Zviran, Thesis Advisor

[illegible]
William J. Haga, Second Reader

[illegible]
David R. Whipple, Chairman
Department of Administrative Sciences

ABSTRACT

This Thesis reviews the increasing need for security in a personal computer (PC) environment and proposes a new approach for securing PC applications at the application layer. The Relay Race Approach extends two standard approaches: data encryption and password access control at the main program level, to the subprogram level by the use of a special parameter, the "Baton". The applicability of this approach is demonstrated in an original Basic application and an existing Dbase IV application, representing both third generation language (3GL) and fourth generation language (4GL) environments. The Approach can add to overall network security in the PC LAN environment as well. The Approach is successful and proposed enhancements can strengthen the Approach.

TABLE OF CONTENTS

I. INTRODUCTION

The proliferation of information systems in virtually all areas of business and government has increased the importance of computer security issues. As more people become computer literate, the risks of ill-intentioned individuals obtaining unauthorized access or violating the integrity and validity of data grow. Potential solutions to computer security risks are varied and numerous because different types of computer hardware, operating systems, and application software have different security strengths and weaknesses.

Different environments and applications require varying levels of security and security measures. Some environments need to target their security measures toward threats of accidental data corruption while others are primarily concerned with unauthorized access to sensitive information. Another computer security issue is system protection from viruses, worms, trojan horses, etc.

Widespread use of personal computers and growth of end user computing have introduced myriad security concerns. Almost every personal computer user is likely to view virus protection, data backup, floppy disk control, and data encryption as primary computer security issues (Murray 1989, Stephenson 1989, Brown 1989). However, many personal computer security concerns for the most part remain largely unaddressed at this time (Pfleeger, 1989). Moreover, the growing population of knowledgeable personal computer users increases the numerical chances of security breaches involving personal computers.

This research explores the unique security issues involving personal computers and proposes a new approach for securing personal computer applications and data.

II. SECURITY IN PERSONAL COMPUTERS

The use of personal computers, including microcomputers, office automation workstations and intelligent workstations, has spread substantially in recent years. Since the introduction of these systems, in the late 1970s, they have undergone far reaching changes and improvements which have brought them almost to the level of performance of large computers (Giladi and Zviran, 1989). Analysis of the development and characteristics of personal computers and large systems shows that the processing speed of present personal computers is equal or even superior to that of the main large systems that were in use during the late 1970s (e.g., IBM 370 series).

The basic security problems for personal computers are the same as those for every other computing environment: applications require secrecy, integrity and availability applied to programs and data. However, security problems of personal computers are more serious than those of mainframes or mini computers due to the lack of security tools and mechanisms. Many of the hardware and software facilities important in assuring security are inappropriate and unavailable in the personal computer environment.

The security problem of personal computers is becoming even more meaningful as these machines are being integrated into computer networks. While many personal computers are being used in a stand-alone mode, others are being connected to networks as front-end terminals and processors, becoming a weak link in the network security chain. This problem becomes more crucial in the open system interconnection (OSI) environment. As the goal of an OSI environment is approached it becomes easier and more economical to connect computers and share resources. Logically, more PCs will be integrated into network systems. As a result, national organizations as well as users are becoming concerned with the vulnerability of personal computers (NCSC, 1985; NTISSC, 1987; Post and Kievit, 1991).

As the name implies, personal computers were initially envisioned as being used by one person. Simple physical security measures would supply the necessary measure of security. This single user view is evident in the design of the popular personal computer operating system, MS-DOS. In most organizations today however, PCs are not personally allocated (Gogan, 1991). In view of this, more security is sometimes required. There is a definite lack of tools to provide security for personal computers.

As the popularity and power of personal computers grows, more people want and obtained access to them. Personal computers distribute computing power to virtually all physical locations within an organization, unlike large machines. For the first time the computing power is not under the control of computer professionals. Persons responsible for mainframe or mini-computer security have limited control over how personal computers were being used within organizations. Generally, personal computer users lack the sensitivity toward computer security issues exhibited by mainframe and mini computer operators. The personal and organizational computer security mechanisms evident in large systems are not automatically in place for personal computers.

A. HARDWARE

The first IBM personal computer was built around the 8088 processor. This processor had no protection scheme. All memory locations were open and unguarded. There were no privileged instructions available only to the operating system or trusted kernel. The newer 80286 and 80386 CPUs have stronger protection capabilities but the MS-DOS operating system is not capable of exploiting them (Post, 1991; Pfleeger, 1989).

Common hardware add-on security measures include physical security measures, security modules, and locks and keys. Each provides various degrees of security against certain types of threats while exhibiting weaknesses against others.

Locking doors to rooms containing computers is effective but in most cases not feasible. Too often it is necessary to allow open access to a room containing the computer. Disconnecting and locking the computer's keyboard in a desk drawer or cabinet provides good security without limiting access to an office space. Unfortunately, physical security measures limit access to all of the computer's programs, not just the sensitive ones. This weakness can lead to under-use of computer assets.

Security modules are expansion boards which plug into industry standard slots on personal computer motherboards to provide security. They usually perform in concert with software utilities. Security modules usually prevent booting from other than the fixed hard disk drive. This ensures that access control software stored on the fixed disk is run upon boot-up. Because the modules must plug into standard slots for compatibility reasons it would be easy for an intruder to locate and remove them. Many casual personal computer users possess sufficient knowledge to quickly open a computer's case, identify specific expansion boards and remove the security module (Stephenson, 1989; Zarger, 1988).

Key type locks coupled to power switches are often used in personal computers as security measures. These locks are an "all or nothing" device. Those who have a key have access to all programs and data and those who do not have a key have access to nothing. They cannot provide universal access to public applications and provide security for private programs and data. Additionally, locking power switches can be defeated quite easily if the computer is housed in a standard case. Once the standard "easy access" case is opened, it is a simple matter to "hot wire" the switch to defeat the lock.

Hardware security solutions are enhanced when cases and fasteners are used which are non-standard and require special tools or keys for access or removal. Additionally, epoxy coatings are useful in protecting hardware items from tampering measures such as hot wiring switches. As with most security issues, using optimum combinations of security measures greatly enhances personal computer security effectiveness.

B. SOFTWARE

Any computer system has, usually, two different types of software: an operating system and application programs. The operating system consists of the system programs, command interpreter, and utilities. The operating system is the focal point for exploring security issues. Application programs are those which accomplish processing desired by the computer user. Application programs make calls to, or use, the operating system to accomplish lower level tasks.

1. Operating Systems

The operating system is the inner-most software layer of a computer system according to the "virtual machine" model (Tanenbaum, 1990). It accomplishes tasks for users and/or the application programs and shields them from complex hardware details. Transparent to users and applications, the function of the operating system is to present the user with the equivalent of an extended machine or virtual machine that is easier to program than the underlying hardware. Its primary task is to keep track of resource usage, to grant resource requests and account for their usage, and to mediate conflicting requests from different programs and users (Tanenbaum, 1987).

At their advent, personal computers were initially equipped with 4 KB of main memory. The operating system had to be small enough to be loaded into this small memory space and still leave room for an application program to run. The early developers of personal computers and their operating systems did not expect these machines to grow in popularity as they have. The operating system was written to provide compactness and functionality in a "personal" environment. This meant one user, one program at a time. Under MS-DOS, anyone with basic knowledge can access and/or change any file or memory location. The current trend is toward personal computer power houses shared by several workers able to run several applications simultaneously utilizing up to 8 MB of main memory. With multiple users instead of the envisioned personal use, MS-DOS does not provide any measure of security. In examining MS-DOS it is clear that it has limitations which cripple its capability to grow into a full-fledged operating system capable of supporting and managing systems which are now in demand.

MS-DOS's major limitation is that when conceived, it allotted only enough bits in its address format to access a maximum of 640 kilobytes of main memory directly. This limit remains in place today because of market pressures for downward compatibility. The most powerful applications programs tend to use most of the 640 kb of memory leaving only enough for the underlying operating system. To install security mechanisms in MS-DOS would undoubtedly reduce the memory space available for use by application programs to an even lower value. It seems that the market pressure for freeing up memory for applications is far greater than any pressure to add security functions to MS-DOS.

Although most operating systems for large systems provide adequate security functions, MS-DOS continues to serve as the personal computer standard with virtually no security capability. Market pressure for compatibility and maximum application space will defeat any move to retrofit MS-DOS with security functionality.

2. Utilities

Utilities are separate system programs that accomplish tasks for users. Their normal function is system management. Since they are optional, commercial software programs are not written to use them. Utility programs are very important in security of personal computer systems. Because the operating system has no security capability, personal computer users often use utility programs to protect their data and programs. There are several different ways in which utility programs are commonly used for security in the personal computer environment. These include encryption of data, password hard disk drive locks with or without hardware locks, and disk residue eliminators. The best commercially available solutions include elements of all three (Stephenson, 1989).

Encryption of data using utility programs provides excellent security of data. The application program can be run by intruders but the data they receive will be nonsense unless first decrypted. Encryption and decryption can be accomplished automatically using batch command files. There are two limitations which come to mind in using data encryption utilities. Data file encryption and decryption are disk intensive activities and consequently are very slow. Additionally, simply securing the data does not keep intruders from running the application program. It simply keeps the intruder from understanding the data. In some cases it may be desirable to ensure intruders are unable to run the application program at all.

Password hard disk drive boot locks are programs which require password authentication to boot and subsequently access the hard drive. They are fast, compact and work well against casual, novice intrusion attempts. Without hardware enhancements, however, they can be bypassed if the intruder boots the computer from a bootable MS-DOS floppy (Stephenson, 1989).

Additionally, access to even non-sensitive programs on the protected system requires password authentication. This limits the use of computer resources to trusted password holders only. In many cases it is desirable to secure only a portion of the functions the personal computer helps perform.

Other utilities rid secondary memory of residue. When files on personal computers are deleted their data remains. The operating system simply deletes the file from the directory, rendering it unlocatable. Intruders can read or copy portions of the memory media in search of sensitive data. Simply deleting files does not protect the information. Utilities such as Norton's *wipe disk* and *wipe* *file* rewrite the disk or file entirely with meaningless data. This destroys all residue left from sensitive files.

These types of utilities are often bundled with hardware which disallows booting from any disk except the one protected by the software.

3. Applications

Application programs are the outer layer of software in the virtual machine model. The application software is a program which interfaces with the user and ensures that the tasks the user wishes to accomplish are completed. The application program makes calls to the operating system to accomplish low level tasks in order for the application to accomplish tasks initiated by the user. The application software is shielded from hardware details by the operating system.

The operating system, MS-DOS, provides no security capability and utilities leave possible back doors and require password access procedures for all applications. If application programs provide their own security capability only programs which require security would require passwords for access. Moreover, common back doors associated with security utility programs are closed to intruders when application programs contain protection schemes. Application programs that need no protection are not limited by running under a larger, hypothetical, security-capable operating system which would use more of the 640 kb main memory than the unprotected MS-DOS. A minor drawback to applications providing their own protection is that the consequential increase in program size occurs in each secured application program. This is a minor drawback as the additional required disk storage space would small. The additional RAM would be required only by programs needing protection, thereby freeing maximum main memory for larger unprotected programs.

C. DATA

Two views of data security prevail: protection against inadvertent data loss and protection of unauthorized access to sensitive data. Inadvertent data loss is a problem of valuable, but not necessarily sensitive, data (Mensching and Adams, 1991). Procedures for precluding inadvertent data loss have been common knowledge since the personal computer's inception and will not be addressed here. Since the operating system as described previously provides no built-in file protection measures, data file encryption must be used to secure data in personal computers.

Utilities are commonly used to encrypt and decrypt data files to ensure protection of sensitive data. Some hardware add-on boards also possess the capability to automatically encrypt and decrypt data files. There are many different algorithms to encrypt and decrypt data. Some of which are considered to be safer than others. The Data Encryption Standard (DES) is the most common one, initially developed for the U.S. government for use by the general public.

D. PERSONNEL

Sensitivity to security issues and an attitude of responsibility on the part of all users in a personal computer environment are necessary for other measures to succeed in providing security. Whereas mainframes and other large systems have separate locked rooms and expert operators shielding them, personal computers are vulnerably distributed throughout an organization. For instance, no security system can succeed if a user leaves the area while a sensitive application is running. No matter how strong the security system, it is useless unless personnel have a healthy attitude toward security and are sensitive to possible threats (Pfleeger, 1989).

III. THE RELAY RACE APPROACH

A. INTRODUCTION

In view of the personal computer operating system's inability to provide security and the limitations associated with security modules and utilities, it becomes worthwhile to explore new techniques for securing individual application programs. Three major threats to the security of an application can be countered, to include unauthorized execution of the main program, data disclosure and unauthorized execution of parts of the program by executing subprograms directly. Traditional methods cover encryption of data files and securing main programs, while the Relay Race Approach extends protection to the subprogram level.

1. Application Access Control

Personal computers are often used by different individuals running different application programs (Gogan, 1991). In most cases, all applications are stored on the same hard disk drive. Allowing access to certain programs by certain individuals while limiting access to valid users of other protected programs stored on the same disk is no trivial task in the PC environment. Since the MS-DOS operating system provides no security kernel, the solutions must be coded into the application programs. Each application program must check for access authorization and take required measures to secure itself against intrusion. This is usually accomplished by an application-oriented Password checking scheme which protects the application at the main menu level.

2. Data File Security

Intrusion is usually for the purpose of achieving access to the system data. One intrusion technique is to bypass the application programs entirely and attempt to gain access to the system data files directly. An intruder could simply browse the file or copy it for later examination at another computer. To overcome this problem, data files must be encrypted.

3. Intra-application Controls

The growth in application software capabilities and the consequent growth in size has dictated that applications be designed as a collection of programs. In such a scheme, a main program calls on subprograms to accomplish specific tasks in support of the system. The main program can be secured with a password-checking scheme to prevent its unauthorized execution. However, access to functions and data can sometimes be achieved by executing subprograms directly without the main program, as depicted in Figure 3.1. To preclude this type of intrusion some method of ensuring that all subprograms are called by their proper calling programs or subprograms must be devised. The

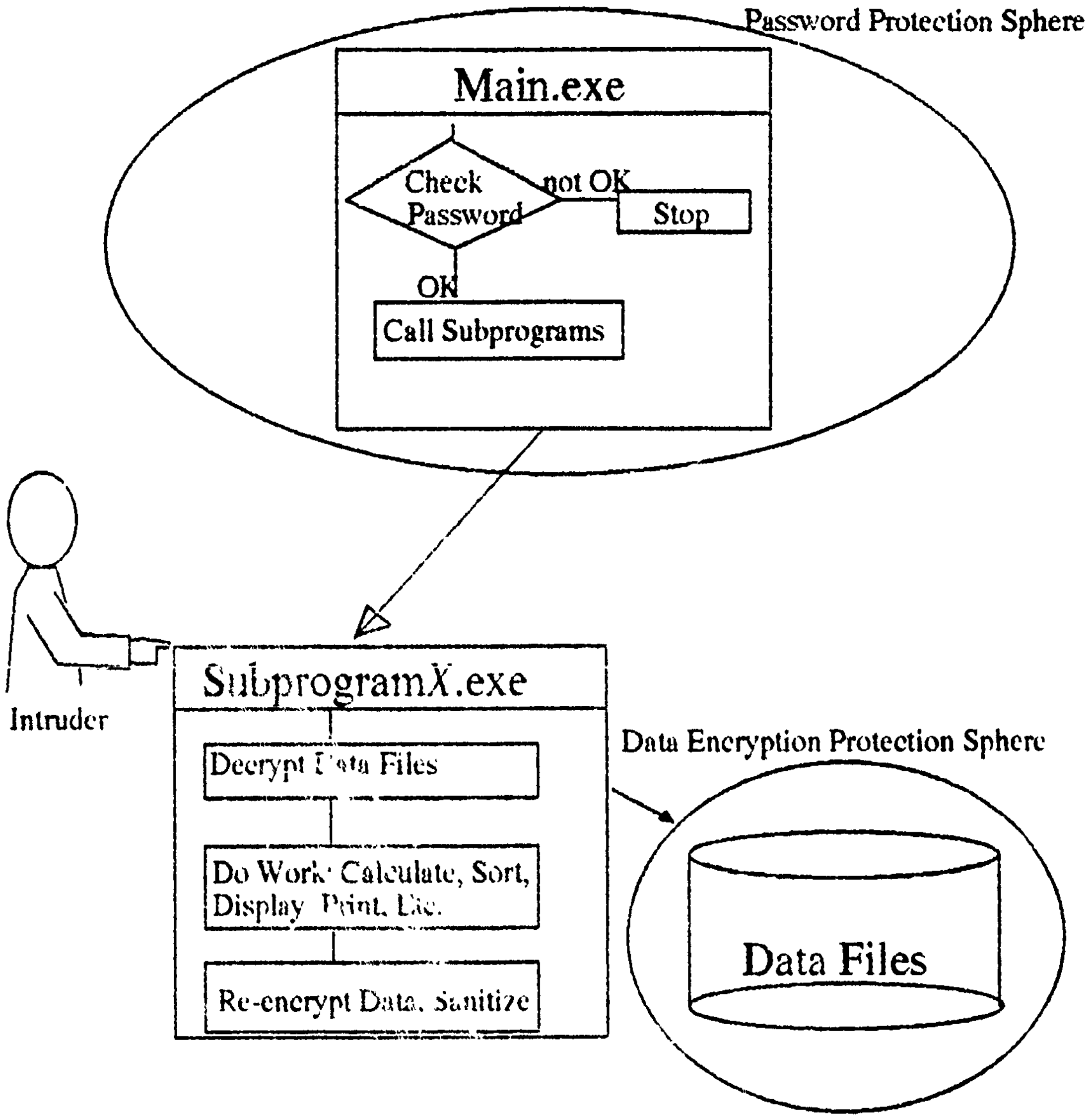
Password Protection Sphere
Main.exe
Check Password
not OK
Stop
OK
Call Subprograms
Intruder
SubprogramX.exe
Decrypt Data Files
Do Work: Calculate, Sort, Display, Print, Etc.
Re-encrypt Data, Sanitize
Data Encryption Protection Sphere
Data Files

Figure 3.1: Password and Encryption Spheres of Protection

approach explored in this research will be called the relay race approach.

B. IMPLEMENTATION CONCEPTS

In order to counter the three threats three methods of protection are implemented in The Relay Race Approach. The first two measures are commonly used in the personal computer environment in an attempt to secure applications. One is basic password checking upon execution ensures user authorization, and the second employs automatic data encryption, decryption and deletion preclude theft of raw data files. However, to preclude program execution via an unprotected subprogram, all subprograms will check for a parameter which can only be valid if the subprogram was called via the main program as illustrated in Figure 3.2. This is a unique measure applied to individual application programs and it is from this third measure that the approach receives its name. In much the same way relay racers must pass a baton or be disqualified, subprograms must receive a certain parameter and pass it to subsequent subprograms or the program execution will be halted by the security system.

1. Password Storage and Management

There are two methods of storing valid passwords to be used by the system to authenticate users: including valid passwords in program source code and storing valid passwords

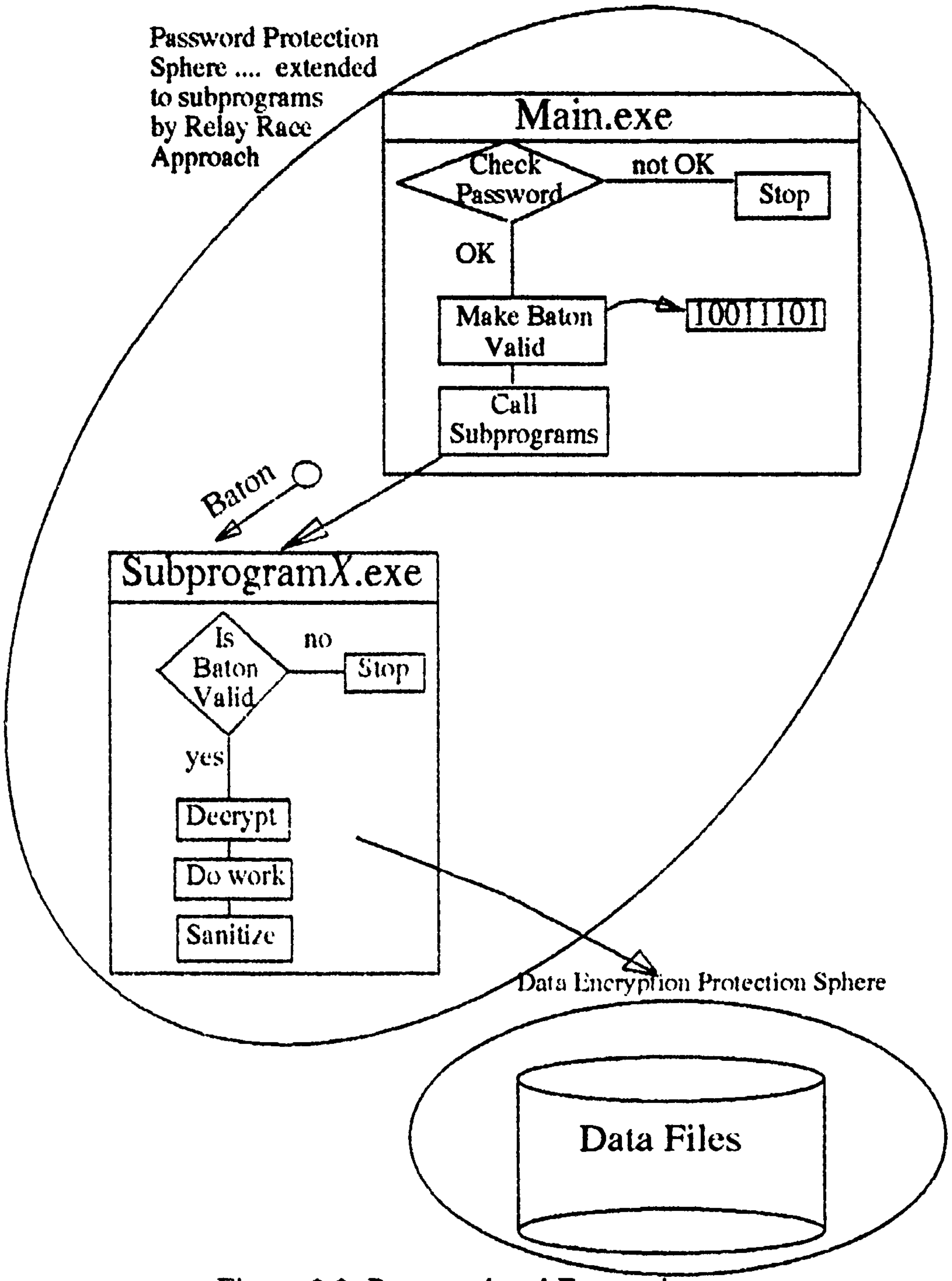

Figure 3.2: Password and Encryption Spheres of Protection extended to subprograms using the Relay Race Approach.

in encrypted data files. Including passwords in source code provides simplicity and security but requires recompilation for each password change. Using encrypted data files containing valid passwords precludes requirement for source code dissemination to user/ administrator but requires thoughtful implementation to ensure security. An intruder could encrypt his/her own password file with a different key and replace the real password file with his/her version (same filename). In order to defeat this intrusion scheme the system must check to determine whether the password file is real or one planted by an intruder. The valid password file will contain a password to be checked against one in the compiled code. The intruder's file would not work if it did not contain this file checking password. A combination of both encrypted data file and compiled password ensures security and precludes source code dissemination and recompilation for routine password changes.

2. Baton and Baton Passing

In order to ensure that subprograms are executed only when called by proper calling programs a global variable, or parameter, can be set upon password authentication and passed from the main program to the called subprograms. Subprograms can, in turn, pass the same parameter to any subprograms they call. Each subprogram can begin execution by checking this parameter before executing further and halt processing if the parameter is invalid. This is analogous to a relay race at a track meet. Without the baton being properly passed and received the relay team cannot complete the race.

3. **Data File Encryption**

Two types of encrypted data files are required for the relay race baton scheme: password file and data storage files. The password file is decrypted, and the decrypted file is then read and deleted. The decryption process leaves the encrypted file intact so that when the system deletes the decrypted files, the original encrypted password file remains for use in future access attempts. Data storage files must be decrypted for reading and recrypted if new data is added or other changes are made. Once again all files decrypted during a process need to have the decrypted copy deleted as soon as possible after they are re-encrypted.

IV. APPLICATION IN A THIRD GENERATION LANGUAGE ENVIRONMENT

A. ENVIRONMENT DESCRIPTION

A simple test application was developed in compiled MS-BASIC. BASIC was chosen as the third generation language for a prototype due to its relatively low power and programmers' wide exposure to it. If the relay race scheme can be implemented in BASIC, it is reasonable to assume that it is possible to implement it in any of the known third generation languages.

B. APPLICATION DESCRIPTION

The prototype application is a simple, menu-driven, maritime minefield planning program designed to minimize the necessity for accurate small scale plotting on geographic charts. The program has options to input planning data, calculate mine drop instructions, save instructions to disk, and print instructions. The application programs and data are protected using The Relay Race Approach.

The MS-DOS directory presentation for the application is provided in Figure 4.1. BASRUN20.EXE is a runtime package required for applications compiled separately such as the minefield planning application. MFPLAN.EXE is the main program containing password checking code and opening menu. The remaining .EXE files are subprograms which accomplish the application's tasks. $ED.MNQ and $ED.NMQ are encrypted data and password files respectively.

```
$ED       MNQ      560  5-19-91  9:24p
$ED       NMQ      128  2-10-91  6:57p
BASRUN20  EXE    63046  6-25-85  4:42p
MFPLAN    EXE     3415  5-19-91  7:55p
MINECALC  EXE     4615  5-19-91  7:56p
MINEPRNT  EXE     2503  5-19-91  7:57p
MINESAVE  EXE     2887  5-19-91  7:56p
MLRETREV  EXE     2279  5-19-91  7:57p
```

Figure 4.1: MS-DOS Directory presentation of the application

The threat of intrusion via subprogram defeating the password authorization and data encryption without baton passing is illustrated in a structure chart of an intrusion attempt (Figure 4.2). When unprotected, an intruder needs only to write a small BASIC program to call MLRETREV.EXE and MINEPRNT.EXE in order to gain access to the system's sensitive data. By combining password checking, data encryption and the Relay Race Approach, this intrusion is thwarted (Figure 4.3).

C. TRANSFORMING CONCEPTS TO CODE

1. Handling Passwords and the Baton

The first operation the scheme must accomplish is password checking. This operation is be accomplished as early as possible in the application. Figure 4.4 contains

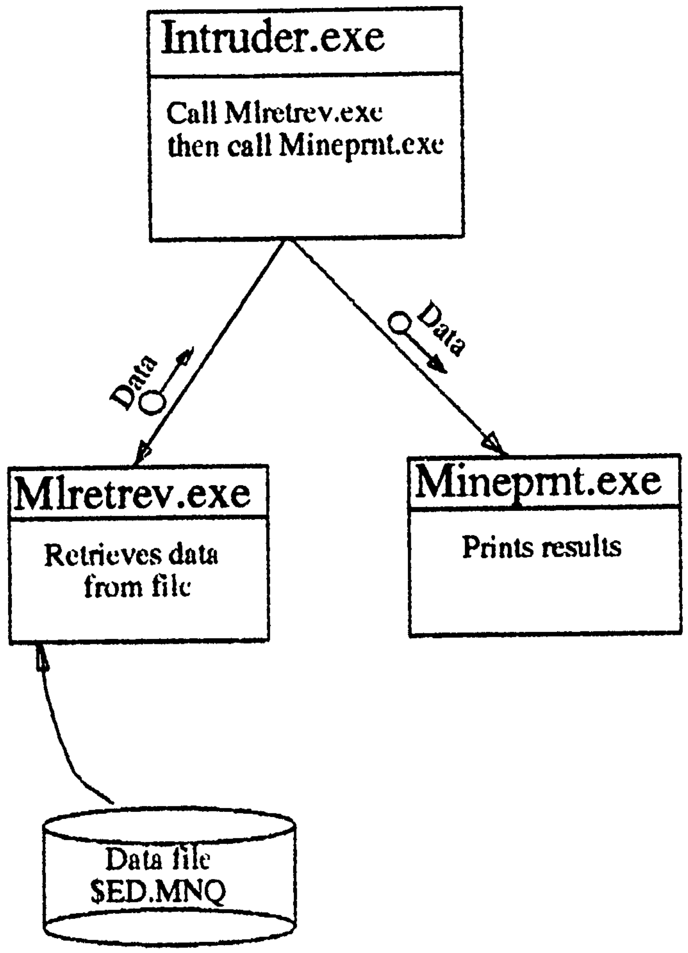

Figure 4.2: Structure chart illustrating how an intruder's program could call subprograms and achieve data access.

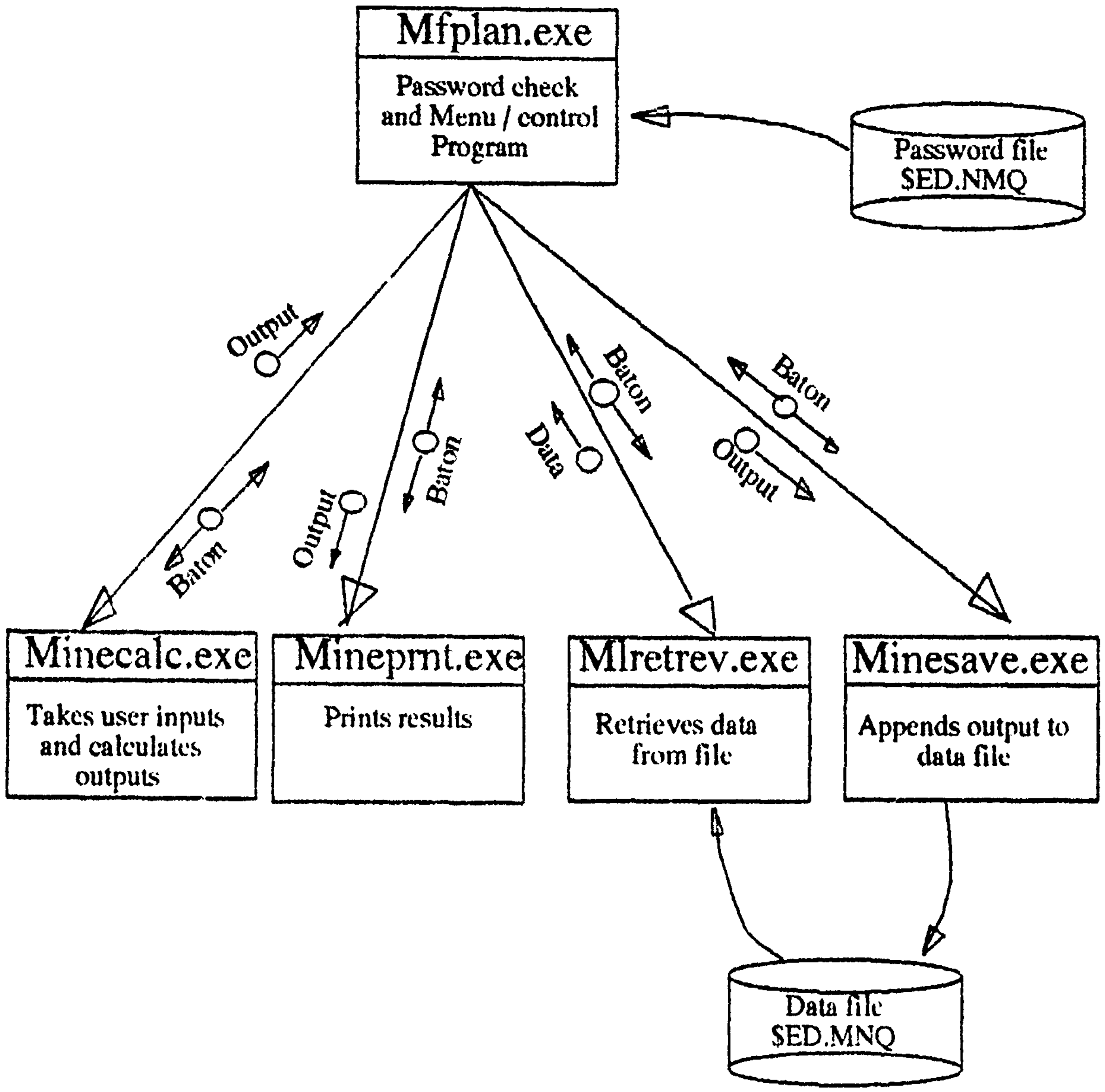

Figure 4.3: Structure chart of Mine - field Planning Program

the required BASIC source code to handle passwords and password checking and initialize the security baton. Line 9 allows the program to be recalled from subprograms without requesting a password each time the main menu appears. Lines 40, 50 and 60 blank the display screen for password entry, input password and return normal function to the display screen. Line 70 creates a decrypted copy of the valid password file and names it "PWORD.DAT". During the execution of the "RCRYPT" program, the user will be prompted to enter an encryptation key twice. Line 80 opens the PWORD.DAT file for input. Lines 90, 100, 110 initialize several variables to be used: N, a loop counter; FOUND$, a flag indicating wether a password is found to be valid or not; and BATON$, the global variable or parameter passed to subprograms to verify that access authorization has been checked prior to subprogram execition. If the password file is found to be empty, line 120 will call the violation routine, (lines 220-290). Lines 140 and 150 input and check the first entry in the password file and ensure it is "scud". This defeats intruders who might plant their own encrypted password file in place of the original. If an imposter password file is detected the violation routine is run. Lines 160-200 are the password checking loop where the input password (PASSWORD$) is checked against each valid password in the file (VALIDPWORD$(N)). If end of file (EOF) is reached without a match the violation routine is run. If a match is found, lines 300-320 are run in order to close the password file, delete it and set the security baton (BATON$) valid. This allows subprograms to be called and run. The violation routine (lines 220-290) also closes and deletes the password file. Lines 240-260 provide a pause situation allowing displayed text message to be read by users before continuing program execution. Lines 350 to 450 represent location of functioning non-security related application code.

```
9 IF BATON$ = "VALID" then GOTO 330
10 LOCATE 13,10
20 PRINT "Enter your password and press ENTER."
30 LOCATE 15,15
40 COLOR 0
50 INPUT PASSWORD$
60 COLOR 7
70 SHELL "RECRYPT $ED.NMQ PWORD.DAT"
80 OPEN "PWORD.DAT" for INPUT as #1
90 N = 0
100 FOUND$ = "F"
110 BATON$ = "INVALID"
120 IF EOF(1) GOTO 220
140 INPUT# 1, FILECHK$
150 IF FILECHK$ = "scud" then GOTO 160 else GOTO 220
160 IF EOF(1) GOTO 220
170 N = N + 1
180 INPUT#1, VALIDPWORD$(N)
190 IF PASSWORD$ = VALIDPWORD$(N) THEN FOUND$ = "T"
200 IF FOUND$ = "T" then GOTO 300 else GOTO 160
210 LOCATE 17,10
220 PRINT "Security Violation!"
230 LOCATE 19,10
240 PRINT "Press any key to continue."
250 A$ = INKEY$
260 IF A$ = "" then 250
270 CLOSE 1
280 KILL "PWORD.DAT"
290 GOTO 500
300 CLOSE 1
310 KILL "PWORD.DAT"
320 BATON$ = "VALID"
330 ( M I N E F I E L D
400   P R O G R A M
450     B O D Y )
500 END
```

Figure 4.4: Code required to check user's password and set "baton" variable

2. Called Subprogram Requirements

Subprograms require very little additional code to accomplish the relay race scheme. As the baton is passed by the COMMON mechanism (sharing variables and values among programs), a simple check of the security baton (BATON$) must be made before each program execution. If the value passed by this variable is valid, execution continues. If the value passed by this variable is found to be invalid, it means that the subprogram was called without valid password authentication. A violation routine is run and the program is aborted. Required source code for subprograms is presented in Figure 4.5.

```
3 OPTION BASE 1
4 DIM YTD(10),TTD(10)
5 COMMON BATON$,NA$,LAH$,IPLAD,IPLAM,IPLAS,LOH$,
  IPLOD,IPLOM,IPLOS,SPD,TRK,N,YTD(),TTD()
20 IF BATON$ = "VALID" GOTO 30 ELSE GOTO 60
30 ( P R O G R A M
40   - - - - - -
50     B O D Y )
60 LOCATE 17,10
70 PRINT "Security Violation!"
80 LOCATE 19,10
90 PRINT "Press any key to continue."
100 A$ = INKEY$
110 IF A$ = "" then 110
120 END
```

Figure 4.5: Code required in subprograms

3. Data Encryption, Decryption and Access Requirements

a. Data Encryption Utility

The encryption program used in this prototype is RCRYPT.COM, an MS-DOS utility. Many different data encryption utilities are available and most will work within this scheme. The application may need to be modified slightly depending on whether the encryption utility requires the key to be entered on the command line as a parameter or prompts the user for the key during execution. The RCRYPT.COM utility in the prototype prompts for the key during execution.

b. Data File Manipulation

This prototype uses one password file and one data file. A flat file of records is used because data for this application is small and response time is not a critical issue. AS shown in Figure 4.6 most of the data manipulations focus on decrypting and reading data. One subprogram (MINESAVE) allows for appending data to the data file. This case requires decrypting the data file, appending new data to the file and re-encrypting the file. The source code required for this operation is presented in Figure 4.7.

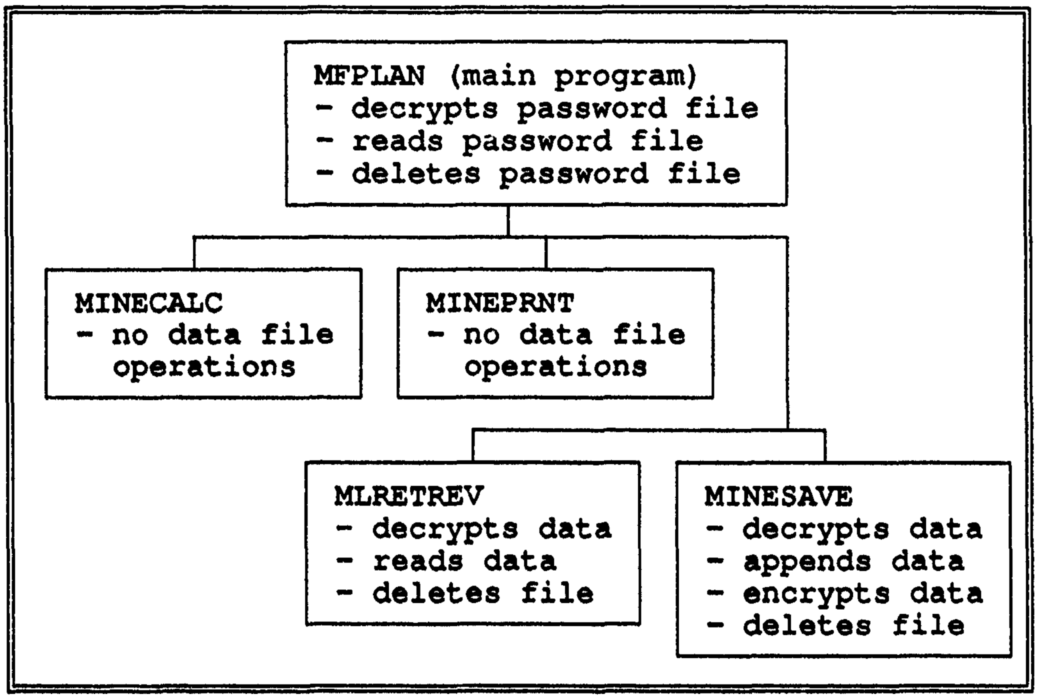

Figure 4.6: Description of data file manipulations for each of the programs in the Basic prototype application.

```
40 PRINT "What name would you like to call the data?"
50 INPUT NA$
60 SHELL "RCRYPT $ED.MNQ MINE.DAT"
70 OPEN "MINE.DAT" for APPEND as #2
80 WRITE #2,NA$,LAH$,IPLAD,IPLAM,IPLAS,LOH$,IPLOD,
  IPLOM,IPLOS,SPD,TRK,N
90 FOR B = 1 to N
100 WRITE #2,YTD(B),TTD(B)
110 NEXT B
120 CLOSE #2
130 SHELL "RCRYPT MINE.DAT $ED.MNQ"
140 KILL "MINE.DAT"
```

Figure 4.7: Code required for data file manipulation in MINESAVE subprogram

c. *System Administration*

The application requires a system administrator to accomplish certain tasks. These tasks include steps to start the system, accomplishing data file housekeeping and changing passwords. Since it is not desirable to supply source code to all users, the application deliverables should include information indicating what the first entry in the password file needs to be. This entry should be unique or nearly unique among different copies of the application to preclude one systems administrator from intruding into another's copy. For example, line 150 Figure 4.4 character string "scud" (FILECHK$) should be identified as the required first entry in the password file and should be different for each copy of the application. To start the system the administrator should add his/her desired passwords, nine at most, to the required first entry, encrypt the file with the desired case sensitive key and the name "$ED.NMQ" and delete the un-encrypted copy of the password file.

The application should also include a data file with one set of test data included to preclude the system from attempting to decrypt and append to an empty file. A copy of this original data file should be maintained by the administrator and used for data housekeeping operations. The data file, like the password file, needs to be encrypted and named in accordance with lines 130 and 70 of Figures 4.7 and 4.4 respectively.

Changing the passwords should be done regularly in any system and should be easy to accomplish so as not to discourage changes when needed. To change passwords, run RCRYPT.COM directly on the $ED.NMQ file and edit the file with new passwords. The required first entry of the file should not be changed or the system will reject the new password file as bogus. Re-encryption of the password file using a new encryption key is needed. Changing the key each time passwords are changed maximizes security.

V. APPLICATION IN A FOURTH GENERATION LANGUAGE ENVIRONMENT

A. ENVIRONMENT DESCRIPTION

The Relay Race Approach was installed into a previously implemented DBaseIV database application. DBase was chosen because of its widespread familiarity and its non-procedural nature. If the approach could be easily grafted into an existing DBase IV generated application, it would be an effective approach for securing other existing applications.

Fourth generation languages are often used in environments where end users build applications. Security may not be considered when users create applications. The Relay Race Approach shows promise as an efficient security measure for these existing end user applications.

The DbaseIV application generator allows users or developers to create fully functional menu driven database applications with little or no coding. Database structures, forms, reports and queries are created using user friendly graphical interfaces and then are combined to work together by the application generator. The application generator generates source code with comments which is compiled into object code that can be run either in the DBase IV environment or with a run-time module.

B. APPLICATION DESCRIPTION

The application is the user version of an automated dive log. It is used for users to enter SCUBA diving events, and query reports such as logs or qualification reports from the database. There is another version which accesses the same database which is used by the system administrator for marketing and other business and organizational functions.

The application accesses four database files: DIVER.DBF, SITE.DBF, DIVE.DBF and QUAL.DBF. It uses one data entry form file, DIVEFORM.SCR. Two query (.QBE) files were slightly modified for use: JOIN1.PRG and QUALLIST.PRG. Two report files were built and used: LOG_REPO.FRM and QUALRPT.FRM. Finally, the application generator created two program files: DLUSER.PRG and USERBAR.PRG.

Since the Relay Race Approach depends on passing parameters between programs, the structure of the application must be understood before the approach can be installed into an existing application. Since the source code was 95 percent generated by DbaseIV the application must be reverse engineered, yielding a structure chart needed for understanding. Figure 5.1 is the structure chart for the application. Only JOIN1.PRG and QUALLIST.PRG can access the data, so only procedures which can possibly call them need to have the additional source code installed.

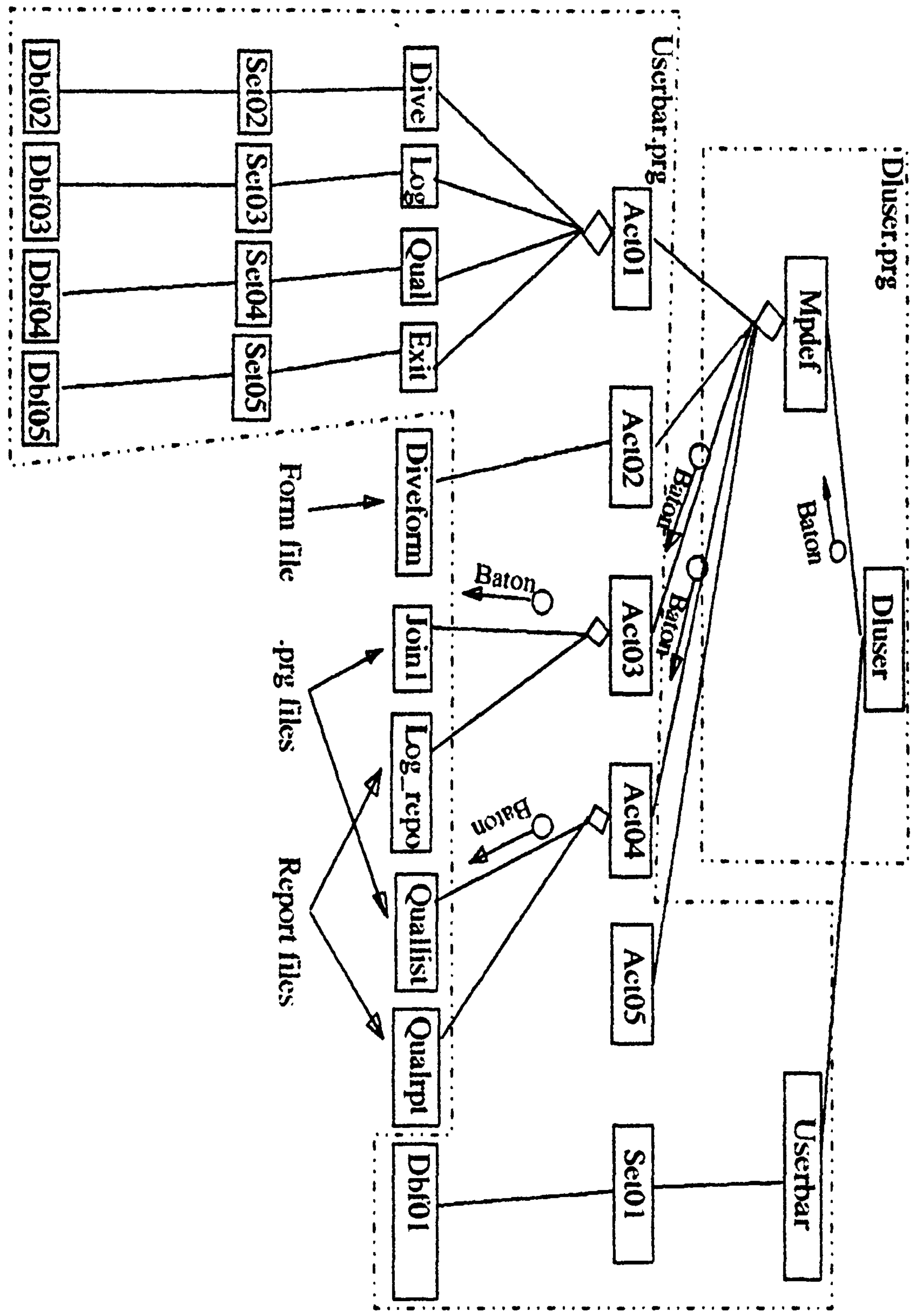

Figure 5.1: Structure Chart of Dive Log Application

These are DLUSER and MPDEF in DLUSER.PRG file, ACT03, ACT04 in USERBAR.PRG file, and JOIN1.PRG and QUALLIST.PRG files.

C. TRANSFORMING CONCEPTS TO CODE

1. Handling Passwords and the Baton

Checking password validity is accomplished first and the code required for this was inserted into the main program, DLUSER.PRG. Figure 5.2 shows the additional source code inserted at the very beginning of the DLUSER.PRG file. The set color commands ensure that the password is not echoed to the screen when the user types their's. In order to get the prompt "Enter Password" on the screen and not the password itself, the prompt and the acceptance of the value for variable "PWORD" had to be separated by the set color command. This is why the ACCEPT string is a space. Set color is used again to return the screen to normal. Since most Dbase IV users will have the capability to compile programs, the passwords were compiled rather than stored in an encrypted file. The logic in the IF / ELSE clause is such that if no password is entered, and the error message which occurs is "ignored" by the user, the program VIOLATIO.PRG will be run, not the rest of DLUSER.PRG. Dbase defaults to the first statement when an error is encountered in an IF/ELSE clause and the user selects "IGNORE" at the error prompt. VIOLATIO.PRG displays a violation message and terminates the program. If the password is found to be valid the data files are decrypted. Since the baton in DBase can be a true parameter instead of a shared variable as was the case in BASIC, a variable does not need to be set. When a procedure is called it simply needs to be called with a value which will be checked by the called procedure. Figure 5.3 illustrates the correct syntax for calling programs and procedures with the parameter required.

```
@3,3 SAY "Enter Password: "
SET COLOR OF NORMAL TO B/B
ACCEPT " " TO PWORD
SET COLOR OF NORMAL TO W+/B
IF (.NOT. PWORD="TIGRIS").AND.(.NOT. PWORD="SCUD")
                  .AND.(.NOT.PWORD="BAGDAD")
  DO VIOLATIO
ELSE
  RUN PKUNZIP ADLDATA -sIRAQ
  ** Rest of Program **
```

Figure 5.2: Code required for password checking in the main program for the Dive Log application.

```
** Calling Program or Procedure **
DO MPDEF WITH "GOOD"


** Called Program or Procedure **
PARAMETER BATON
ON ERROR CANCEL
IF .NOT. BATON = "GOOD"
  DO VIOLATIO
ELSE
  ** Rest of Program **
```

Figure 5.3: Code required for calling subprograms and procedures with parameters.

2. Called Procedures or Subprograms Requirements

Called subprograms or procedures which receive the security parameter BATON must contain the PARAMETERS statement as shown in Figure 5.3. It was discovered during testing that if an intruder attempted to call a subprogram or procedure directly without the required parameter, Dbase displays an error message displaying the (IF .NOT. BATON = "GOOD") line of source code and a prompt "PARAMETER NOT FOUND". This would give the intruder information required to successfully call the subprogram or procedure on his next attempt. The "ON ERROR CANCEL" line terminates program execution when any error occurs to remedy the situation. The IF/ELSE clause checks for the security baton and runs the violation procedure or the rest of the program accordingly.

3. Data Encryption, Decryption and Administration.

Since the application uses four different data files the PKZIP/PKUNZIP utility programs were selected for encryption and decryption of data files. It allows for compression and encryption of multiple files into one single file. As depicted in Figure 5.2 the encryption key "IRAQ" is compiled into the program instead of being prompted from and entered by the user.

The procedure ACT05 in the USERBAR.PRG file, (Figure 5.1), is executed to exit the system. Data encryption and residue housekeeping is accomplished here. The required code is shown in Figure 5.4

```
RUN PKZIP ADLDATA -m -SIRAQ *.DBF
```

Figure 5.4: Code required for encrypting data files and removing the decrypted data files.

The system administrator has only to periodically recompile the source code changing passwords and encryption keys. Access to the source code should be limited to trusted personnel only as it contains information which would greatly simplify intrusion.

VI. THE RELAY RACE APPROACH AND LOCAL AREA NETWORK SECURITY

The explosion of personal computers in the workplace has led to the need for data communication and asset sharing among an organization's Pcs. Local area networks (LANs) efficiently provide these attributes and are being utilized extensively today.

A. ELEMENTS AND FUNCTIONALITY OF LANS

LAN implementation includes installing LAN hardware expansion cards in the computers which are to be linked, linking the computers together using a cabling system, and installing a LAN operating system on the machines. One of the machines is designated as the server and the rest are clients. The full operating system resides on the server while only a shell or subset resides on each client. In popular PC LANs the network operating system still utilizes MS-DOS but provides added network functions.

Communication between machines or nodes in a network involves multiple communication protocols. Each protocol level uses functions provided at lower levels by lower level protocols.

B. SECURITY IN LANS

Most LAN operating systems provide security functions capable of multi-level security of files and physical devices. These measures combined with certain physical security measures involving the network server can protect assets against casual intrusion attempts. However, if physical access to the network server can be gained an intruder could attempt to load a different copy of the network operating system onto the server and give himself access to protected files and/or devices. Many LANs place printers and other periphrials along side the server and the server therefore cannot be physically isolated from the users or public.

C. ENHANCING OVERALL LAN SECURITY

Even though network operating systems oftem provide security features, the Relay Race Approach can significantly strengthen overall security. The Relay Race Approach provides efficient security at the application layer complementing security features implemented at the LAN operating system layer. For example, if an intruder were capable of accessing the LAN server, load a different copy of the LAN operating system and attempt to access a protected application, additional security provided by the Relay Race Approach would significantly hamper his attempts. The additional layer of security would most likely end the intruder's attempt: at least for that session. Additionally, combining security measures implemented at the

LAN operating system layer with those at the application layer can reduce requirements for "armor plated" physical security measures such as heavy duty locks, doors or cabinets for the network server.

Both prototype applications were installed on a LAN. Both executed as expected and illustrated feasibility of the Relay Race Approach as a security measure for applications running on LANs.

VII. CONCLUSIONS

Personal computer security is an issue of increasing importance to computer professionals. It is valuable to explore efficient methods of providing or enhancing PC security. The Relay Race Approach provides or enhances security in the PC environment efficiently. The Approach can be strengthened using deceptive measures to thwart intrusions by all but those thoroughly familiar with the application source code.

A. THE NEED

The increased need for PC security is evident in view of several recent trends. First, PCs are being used in an increasing number of different business areas. These include those areas where sensitive processing is common. Secondly, more persons are becoming familiar enough with PC's and MS-DOS to be considered capable of casual intrusion into marginally protected PC environments. Finally, the increase in public sensitivity to privacy of information issues dictates the need for increased security in areas once thought to be of a non-sensitive nature.

B. REQUIRED ATTRIBUTES

For these reasons an approach with the following attributes would be of significant value. It should be compact, as application program size is of great concern in the PC environment. The approach should be flexible or multi-leveled, that is, it should allow public access to some applications and limit access to other application(s) to only their specified set of authorized users. The approach should be easy to implement, even in existing applications. Increasing end-user application development makes this a valuable attribute. The Relay Race Approach exhibits these desired attributes and is strong enough to withstand casual attacks from intruders with strong knowledge of MS-DOS and PCs.

C. POSSIBLE ENHANCEMENTS

The relay approach depends on the premise that an intruder does not have access to the application source code and knowledge of how the approach was implemented in the application. There are two modifications which could enhance security just in case knowledge of the approach and/or application source code is compromised: unique application copies and deceptive and dynamic baton variables. Additionally, disk file residue eliminators could strengthen security.

1. Unique Application Copies

First, it would be important to make different copies of the application utilize unique or nearly unique password files. This would be accomplished by compiling many versions of the program, each using a different first entry in the password file (the password file check variable). This would defeat an intruder who might have one copy of the application and attempt to insert his password file into another system and using it to gain access to the other system's data.

2. Deceptive and Dynamic Batons

To further help deceive intruders who might gain access to the program source code, the "baton" may be concealed. Suppose in the Basic application the baton variable were "MINEDIST#" instead of "BATON" and was of type integer, Figure 4.4. This would slow a potential intruder's conceptualization as he browses the source code in search of security hints. Additionally, dynamic batons can be employed. Such a baton variable can be set to valid indirectly through one or more intermediate variables which might appear to be accomplishing some arithmetic operations. The value given to the baton variable may also change often but retain some characteristic for the validity check. For instance, the baton could change value but retain even divisibility by 17 and the validity check would be designed to test for that.

3. Disk File Residue Eliminators

Finally, using a filewipe type residue eliminating program instead of simple MS-DOS delete command in the application would provide an extra degree of security to counter random disk sector searches.

The Relay Race Approach provides efficient, casual security for personal computer applications in today's environment of increasing PC security Threats.

Appendix A: Source code for Minefield Planning Application in BASIC.

```
1 'MFPLAN.BAS - Prototype 2 4-21-91 of Relay Race Baton PC security system.
3 OPTION BASE 1
4 DIM YTD(10),TTD(10)
5 COMMON
BATON$,NA$,LAH$,IPLAD,IPLAM,IPLAS,LOH$,IPLOD,IPLOM,IPLOS,SPD,TRK,N,YT
D(),TTD()
7 DIM VALIDPW$(10)
8 CLS
9 IF BATON$="VALID" THEN GOTO 140
18 LOCATE 13,10
20 PRINT "Enter your password and press ENTER."
21 LOCATE 15,15
22 '              ****BLACKEN SCREEN TO HIDE PASSWORD AS IT IS ENTERED &
GET PASSWORD
23 COLOR 0
24 INPUT PASSWORD$
25 '                                                        ****RESET
SCREEN
26 COLOR 7
27 SHELL "RCRYPT $ED.NMQ PWORD.DAT" '        ****DECODE FILE OF VALID
PASSWORDS AND CHECK USER'S FOR VALIDITY
28 OPEN "PWORD.DAT" FOR INPUT AS #1
30 N=0:FOUND$="F": BATON$="INVALID"
32 N=N+1
34 IF EOF(1) GOTO 50
36 INPUT# 1, VALIDPW$(N)
38 IF PASSWORD$=VALIDPW$(N) THEN FOUND$="T"
40 IF FOUND$="T" THEN GOTO 60 ELSE GOTO 32
50 IF FOUND$="F" THEN CLS:LOCATE 17,10:PRINT "Your password is invalid,
access denied."
51 '                                                        ****PAUSE TO
READ MESSAGE
52 LOCATE 19,10:PRINT "Press any key to continue."
54 A$=INKEY$:IF A$="" THEN 54
56 CLOSE 1:KILL "PWORD.DAT" '       ****CLOSE PASSWORD FILE & ERASE IT
58 GOTO 240   '                              ****STOP
60 CLOSE 1:KILL "PWORD.DA'.
62 BATON$="VALID":'                          ****BUILD BATON
80 CLS
90 LOCATE 5,5:PRINT "Welcome to Minefield Planning. A simple Basic program to"
100 LOCATE 6,5:PRINT "assist in planning air deployed minefields. Given IP lat"
110 LOCATE 7,5:PRINT "and long, hole lat & long's, track, speed and trajectory"
120 LOCATE 8,5:PRINT "the program will calculate and securely store and/or "
130 LOCATE 9,5:PRINT "print time to drop and distance to drop."
140 LOCATE 11,10:PRINT "MAIN MENU"
150 LOCATE 13,5:PRINT "1 - Enter new data and calculate drops"
155 LOCATE 14,5:PRINT "2 - Retrieve previously stored solution from disk"
```

```
160 LOCATE 15,5:PRINT "3 - Print data from earlier calculated or retrieved line"
165 LOCATE 16,5:PRINT "4 - Save current mineline calculations to disk"
170 LOCATE 17,5:PRINT "5 - EXIT SYSTEM"
180 LOCATE 19,10:PRINT "Enter your choice"
190 INPUT CH$
200 IF CH$="1" THEN CHAIN "MINECALC" ELSE IF CH$="2" THEN CHAIN
"MLRETREV" ELSE IF CH$="3" THEN CHAIN "MINEPRNT" ELSE IF CH$="4" THEN
CHAIN "MINESAVE" ELSE IF CH$="5" THEN GOTO 240
210 CLS
220 LOCATE 10,10:PRINT "ERROR! choose 1, 2, 3, 4 OR 5"
230 goto 80
240 CLS:END
```

```
1 'MINECALC.BAS
3 OPTION BASE 1
4 DIM YTD(10),TTD(10)
5 COMMON
BATON$,NA$,LAH$,IPLAD,IPLAM,IPLAS,LOH$,IPLOD,IPLOM,IPLOS,SPD,TRK,N,YT
D(),TTD()
7 DIM HLAD(10),HLAM(10),HLAS(10),
HLOD(10),HLOM(10),HLOS(10),HLAMX(10),HLA(10)
8 DIM
HLOMX(10),HLO(10),NSDIFF(10),EWDIFF(10),NSYDS(10),EWYDS(10),TOTYDS(10)
9 DIM TOTYDSCHK(10)
10 CLS
20 IF BATON$ = "VALID" GOTO 30 ELSE GOTO 1065
30 cls:LOCATE 2,5:PRINT "Enter Data Below Prompts."
40 LOCATE 4,5:PRINT "Is I.P. Latitude N or S? (CAPITALS)":INPUT LAH$
50 LOCATE 5,5:PRINT "Degrees":LOCATE 5,15:PRINT "Minutes"
52 LOCATE 5,25:PRINT "Seconds"
60 LOCATE 6,5:INPUT IPLAD:LOCATE 6,15
61 IPLADR=IPLAD*3.141593/180
62 INPUT IPLAM:LOCATE 6,25:INPUT IPLAS
70 LOCATE 8,5:PRINT "Is I.P. Longitude E or W? (CAPITALS)":INPUT LOH$
80 LOCATE 9,5:INPUT IPLOD:LOCATE 9,15:INPUT IPLOM:LOCATE 9,25:INPUT
IPLOS
90 LOCATE 11,5
92 PRINT "Enter true track from I.P. to holes in 3 digits(001-360)."
94 INPUT TRK
95 TRKR=TRK*3.141593/180
100 LOCATE 13,5:PRINT "Enter groundspeed in knots.":INPUT SPD
110 LOCATE 14,5
112 PRINT "Enter weapon trajectory in yards for your speed and altitude (from
TACREFMAN)."
114 INPUT TRAJ
120 LOCATE 18,5:PRINT "Enter number of mines in this line.":INPUT N
121 CLS
130 IPLAMX = IPLAM + (IPLAS/60)
131 IPLA = IPLAD + (IPLAMX/60)
132 IPLOMX = IPLOM + (IPLOS/60)
133 IPLO = IPLOD + (IPLOMX/60)
```

```
134 IF (LAH$="N") AND (LOH$="W") GOTO 140 ELSE GOTO 372
140  FOR I = 1 to N
150  LOCATE 2,20:PRINT "Hole ";I
160  LOCATE 3,5
162  PRINT "Latitude: Deg   Min   Sec   Longitude: Deg   Min   Sec"
165  B=4+I
170  LOCATE B,15:INPUT HLAD(I)
171  LOCATE B,21:INPUT HLAM(I)
172  LOCATE B,27:INPUT HLAS(I)
173  LOCATE B,44:INPUT HLOD(I)
174  LOCATE B,50:INPUT HLOM(I)
175  LOCATE B,56:INPUT HLOS(I)
250  HLAMX(I) = HLAM(I) + (HLAS(I)/60)
260  HLA(I) = HLAD(I) + (HLAMX(I)/60)
270  HLOMX(I) = HLOM(I) + (HLOS(I)/60)
280  HLO(I) = HLOD(I) + (HLOMX(I)/60)
290  NSDIFF(I) = HLA(I) - IPLA
300  EWDIFF(I) = IPLO - HLO(I)
310  NSYDS(I) = NSDIFF(I) * 2020 * 60
320  EWYDS(I) = EWDIFF(I) * 2020 * 60 * COS(IPLADR)
325  IF ((TRK>85)AND(TRK<95))OR((TRK>265)AND(TRK<275))THEN 340
330  TOTYDS(I) = (NSYDS(I))/(COS(TRKR))
335   GOTO 350
340  TOTYDS(I) = (EWYDS(I))/(SIN(TRKR))
350  YTD(I) = TOTYDS(I) - TRAJ
360  TTD(I) = ((((YTD(i)/2020)/SPD)* 60)* 60)
370  NEXT I
371 GOTO 820
372 PRINT "NE, SW AND SE HEMISPHERE PROBLEMS ARE NOT IMPLEMENTED
AT THIS TIME."
374 cls:GOTO 1070
820 CLS
830 LOCATE 2,2:PRINT "IP"
831 LOCATE 2,17:PRINT "Track"
832 LOCATE 2,25:PRINT "Speed"
833 LOCATE 2,33:PRINT "Yards to drop"
834 LOCATE 2,49:PRINT "Time to drop"
840 LOCATE 4,2:PRINT LAH$
841 LOCATE 4,4:PRINT IPLAD
842 LOCATE 4,7:PRINT IPLAM
843 LOCATE 4,10:PRINT IPLAS
844 LOCATE 4,17:PRINT TRK
845 LOCATE 4,25:PRINT SPD
846 locate 5,2:print LOH$
847 locate 5,4:print IPLOD
848 locate 5,8:print IPLOM
849 locate 5,11:print IPLOS
850  FOR K = 1 to N
860  L = 3 + K
870  LOCATE L,33:PRINT YTD(K)
872  LOCATE L,49:PRINT TTD(K)
880  NEXT K
```

```
890 LOCATE 18,10:PRINT "Press any key to continue."
900 B$=INKEY$:IF B$="" THEN 900
910 CHAIN "MFPLAN"
1065 cls:locate 10,10:print "Security Violation, Access Denied."
1066 locate 11,10:print "press any key to continue."
1067 z$=inkey$:if z$="" then 1067
1070 cls:END


1 'MINEPRNT.BAS - print module for minefield planning program.
3 OPTION BASE 1
4 DIM YTD(10),TTD(10)
5 COMMON
BATON$,NA$,LAH$,IPLAD,IPLAM,IPLAS,LOH$,IPLOD,IPLOM,IPLOS,SPD,TRK,N,YT
D(),TTD()
20 CLS
30 IF BATON$="VALID" GOTO 40 ELSE GOTO 250
40 LPRINT " ":LPRINT " "
45 LPRINT ,,"UNCLASSIFIED":LPRINT " "
50 LPRINT ,"       Minefield Planning Report"
60 LPRINT ,,"for " NA$
65 LPRINT " "
70 LPRINT "Initial Position:"
80 LPRINT LAH$ " " IPLAD "-" IPLAM "-" IPLAS "  " LOH$ " " IPLOD "-" IPLOM "-"
IPLOS
90 LPRINT " "
100 LPRINT "True Track: " TRK "    " "Aircraft Groundspeed: " SPD
110 LPRINT " "
115 LPRINT "Hole #","Time to Drop ","Distance to Drop:"
117 LPRINT ,"(seconds)","(yards)"
120  FOR K = 1 TO N
130  LPRINT K,TTD(K),YTD(K)
140  NEXT
142 LPRINT " "
145 LPRINT ,,"UNCLASSIFIED"
150 CHAIN "MFPLAN"
250 CLS:LOCATE 10,10:PRINT "Security  Violation, Access Denied."
260 LOCATE 11,10:PRINT "press any key to continue"
270 q$=inkey$:if q$="" then 270
280 cls:end


1 'MINESAVE.BAS
3 OPTION BASE 1
4 DIM YTD(10),TTD(10)
5 COMMON
BATON$,NA$,LAH$,IPLAD,IPLAM,IPLAS,LOH$,IPLOD,IPLOM,IPLOS,SPD,TRK,N,YT
D(),TTD()
20 IF BATON$="VALID" THEN GOTO 40 ELSE GOTO 140
21 CLS
22 LOCATE 2,2:PRINT "When this program stores a file it does not store the"
23 locate 3,2:print "trajectory or the hole lat/long. Therefore no classified"
```

```
24 locate 4,2:print "data is stored or can be derived from the file as long as"
25 locate 5,2:print "the minefield is an exercise field and not a real operational one."
26 LOCATE 7,5:PRINT "Do you wish to store this data in the file? (y or n) "
27 INPUT AN$
28 IF AN$="Y" OR AN$="y" GOTO 40 ELSE IF AN$="N" OR AN$="n" GOTO 130 ELSE
GOTO 31
31 LOCATE 10,5:PRINT "Error choose y or n."
32 LOCATE 12,5:PRINT "Press any key to continue."
33 B$=INKEY$:IF B$="" THEN 33
34 GOTO 21
40 LOCATE 5,5:PRINT "What name would you like to store the data under?"
45 locate 6,8:print "(all lower case and remember it please)"
50 INPUT NA$
60 SHELL "RCRYPT $ED.MNQ MINE.DAT"
70 OPEN "MINE.DAT" FOR APPEND AS #2
80 WRITE #2,NA$,LAH$,IPLAD,IPLAM,IPLAS,LOH$,IPLOD,IPLOM,IPLOS,SPD,TRK,N
81  FOR B=1 TO N
82  WRITE #2,YTD(B),TTD(B)
83  NEXT B
90 CLOSE #2
100 SHELL "RCRYPT MINE.DAT $ED.MNQ"
110 KILL "MINE.DAT"
130 CHAIN "MFPLAN"
140 CLS:LOCATE 10,10:PRINT "Security Violation, access denied."
150 LOCATE 11,10:PRINT "Press any key to continue."
160 p$=inkey$:if p$="" then 160
170 CLS:END


1 'MLRETREV.BAS
3 OPTION BASE 1
4 DIM YTD(10),TTD(10)
5 COMMON
BATON$,NA$,LAH$,IPLAD,IPLAM,IPLAS,LOH$,IPLOD,IPLOM,IPLOS,SPD,TRK,N,YT
D(),TTD()
20 CLS
30 IF BATON$="VALID" THEN GOTO 40 ELSE GOTO 160
40 LOCATE 5,5:PRINT "Enter the name you stored desired data under. (lower case
please)"
50 INPUT NM$
60 SHELL "RCRYPT $ED.MNQ MINE.DAT"
70 OPEN "MINE.DAT" FOR INPUT AS #3
80 IF EOF(3) THEN CLOSE #3:PRINT "   NOT FOUND":KILL "MINE.DAT":GOTO 150
90 INPUT #3,NA$
110 IF NA$=NM$ THEN GOTO 120 ELSE GOTO 80
120 INPUT #3,LAH$,IPLAD,IPLAM,IPLAS,LOH$,IPLOD,IPLOM,IPLOS,SPD,TRK,N
121  FOR C=1 TO N
122  INPUT #3,YTD(C),TTD(C)
123  NEXT C
130 CLOSE #3:KILL "MINE.DAT"
150 CHAIN "MFPLAN"
160 CLS:LOCATE 10,10 PRINT "Security Violation, access denied."
```

```
170 LOCATE 11,10:PRINT "Press any key to continue."
180 s$=inkey$:if s$="" then 180
190 cls:end
```

Appendix B: Source code for Dive Log Application in Dbase IV

```
************************************************************************
* Program......: DLUSER.PRG
* Author.......:        This is an APPLICATION OBJECT.
* Date.........: 8-04-91
* Notice.......: Type information here or greetings to your users.
* dBASE Ver....: See Application menu to use as sign-on banner.
* Generated by.: APGEN version 1.3
* Description..: user application of dive log database.

* Description..: Main routine for menu system
************************************************************************

* ADDED CODE FOR SECURITY MODULE
@3,3 SAY "Enter Password: "
SET COLOR OF NORMAL TO B/B
ACCEPT " " TO PWORD
SET COLOR OF NORMAL TO W+/B
IF (.NOT. PWORD="TIGRIS").AND.(.NOT. PWORD="SCUD").AND.(.NOT.
PWORD="BAGDAD")
 DO VIOLATIO
ELSE
RUN pkunzip adldata -sIRAQ
******************

*-- Setup environment
SET CONSOLE OFF
IF TYPE("gn_ApGen")="U"
  CLEAR WINDOWS
  CLEAR ALL
  CLOSE ALL
  CLOSE PROCEDURE
  gn_ApGen=1
ELSE
  gn_ApGen=gn_ApGen+1
  IF gn_ApGen > 4
    Do Pause WITH "Maximum level of Application nesting exceeded."
    RETURN
  ENDIF
  PRIVATE gn_oldsize
  gn_oldsize=gn_scrsize
  PRIVATE gc_bell, gc_carry, gc_clock, gc_century, gc_confirm, gc_deli,;
        gc_safety, gc_status, gc_score, gc_talk, gc_key, gc_prognum,;
        gc_quit, gc_color, gc_display, gl_color, gl_batch, gn_scrsize
ENDIF
*-- Store some sets to variables
gc_bell   =SET("BELL")
```

```
gc_carry  =SET("CARRY")
gc_clock  =SET("CLOCK")
gc_color  =SET("ATTRIBUTE")
gc_century=SET("CENTURY")
gc_confirm=SET("CONFIRM")
gc_cursor =SET("CURSOR")
gc_deli   =SET("DELIMITERS")
gc_display=SET("DISPLAY")
gc_safety =SET('SAFETY")
gc_status =SET("STATUS")
gc_score  =SET("SCOREBOARD")
gc_talk   =SET("TALK")
SET CONSOLE ON
IF gc_display <> "EGA25"
   gn_error=0
   ON ERROR ??
   SET DISPLAY TO EGA25
   ON ERROR
ENDIF

SET BELL ON
SET CARRY OFF
SET CENTURY OFF
SET CLOCK OFF
SET CONFIRM OFF
SET DELIMITERS TO ""
SET DELIMITERS OFF
SET DEVICE TO SCREEN
SET ESCAPE ON
SET EXCLUSIVE OFF
SET LOCK ON
SET MESSAGE TO ""
SET PRINT OFF
SET REPROCESS TO 4
SET SAFETY ON
SET TALK OFF

*-- Initialize global variables
gl_batch=.F.        && is a batch operation in progress
gl_color= ISCOLOR() .AND. SET("DISPLAY") <> "CGAMONO"
gn_error=0          && 0 if no error, otherwise an error occurred
gn_ikey=0           && keypress returned from the INKEY() function
gn_scrsize=21       && number of lines on screen
gn_send=0           && return value from popup of position menus
gn_trace=1          && sets trace level, however you need to change template
gc_brdr='1'         && border to use when drawing boxes
gc_dev='CON'        && Device to use for printing - See Proc. PrintSet
gc_key='N'          && leave the application
gc_prognum='  '     && internal program counter to handle nested menus
gc_quit=' '         && memvar for return to caller
listval='NO_FIELD'  && Pick List value
```

```
*-- remove asterisk to turn clock on
* SET CLOCK TO

*-- Blank the screen
SET COLOR TO
CLEAR
SET SCOREBOARD OFF
SET STATUS OFF

*-- Define menus
**********************
DO MPDEF WITH "GOOD"      && execute Menu Process DEFinition
********************
*-- Execute main menu
DO WHILE gc_key = 'N'
*********************
   DO USERBAR WITH "BOO","GOOD"
**********************
   IF gc_quit = 'Q'
      EXIT
   ENDIF
   ACTIVATE WINDOW Exit_App
   lc_conf=SET("CONFIRM")
   lc_deli=SET("DELIMITER")
   SET CONFIRM OFF
   SET DELIMITER OFF
   @ 1,2 SAY "Do you want to leave this application?" ;
         GET gc_key PICT "!" VALID gc_key $ "NY"
   READ
   SET CONFIRM &lc_conf.
   SET DELIMITER &lc_deli.
   RELEASE lc_conf, lc_deli
   DEACTIVATE WINDOW Exit_App
ENDDO

*-- Reset environment
DEACTIVATE WINDOW FullScr
?? Color(gc_color)
gn_ApGen=gn_ApGen-1
SET BELL  &gc_bell.
SET CARRY &gc_carry.
SET CLOCK &gc_clock.
SET CENTURY &gc_century.
SET CONFIRM &gc_confirm.
SET CURSOR  &gc_cursor.
SET DELIMITERS &gc_deli.
SET DISPLAY TO &gc_display.
SET STATUS &gc_status.
SET SAFETY &gc_safety.
SET SCORE  &gc_score.
SET TALK   &gc_talk.
```

```
IF gn_Apgen < 1
   ON KEY LABEL F1
   CLEAR WINDOWS
   CLEAR ALL
   CLOSE ALL
   CLOSE PROCEDURE
   SET ESCAPE ON
   SET MESSAGE TO ""
   CLEAR
ELSE
   DEFINE WINDOW FullScr FROM 0,0 TO gn_oldsize+3,79 NONE
   DEFINE WINDOW Savescr FROM 0,0 TO gn_oldsize,79 NONE
   DEFINE WINDOW Helpscr FROM 0,0 TO gn_oldsize,79 NONE
   ACTIVATE WINDOW FullScr
ENDIF
******************
ENDIF
******************

RETURN

******************************************************************************
**
* Description..: Procedure files for generated menu system.
* The programs that follow are common to main routines
* The last procedure is the Menu Process DEFinition
******************************************************************************
**
PROCEDURE Lockit
PARAMETER ltype
IF NETWORK()
   gn_error=0
   ON ERROR DO Multerr
   IF ltype = "1"
     ll_lock=FLOCK()
   ENDIF
   IF ltype = "2"
     ll_lock=RLOCK()
   ENDIF
   ON ERROR
ENDIF
RETURN

PROCEDURE Info_Box
PARAMETERS lc_say
? lc_say
? REPLICATE("-",LEN(lc_say))
?
RETURN
* EOP: Info_Box
```

```
PROCEDURE get_sele
*-- Get the user selection & store BAR into variable
gn_send = BAR()  && Variable for print testing
DEACTIVATE POPUP
RETURN

PROCEDURE ShowPick
listval=PROMPT()
IF LEFT(entryflg,1)="B"
   lc_file=POPUP()
   DO &lc_file. WITH "A"
   RETURN
ENDIF
IF TYPE("lc_window")="U"
   ACTIVATE WINDOW ShowPick
ELSE
   ACTIVATE WINDOW &lc_window.
ENDIF
STORE 0 TO ln_ikey,x1,x2
ln_ikey=LASTKEY()
IF ln_ikey=13
   x1=AT(TRIM(listval)+',',lc_fldlst)
   IF x1 = 0
      lc_fldlst=lc_fldlst+TRIM(listval)+','
   ELSE
      x2=AT(',',SUBSTR(lc_fldlst,x1))
      lc_fldlst=STUFF(lc_fldlst,x1,x2,'')
   ENDIF
   CLEAR
   ? lc_fldlst
ENDIF
ACTIVATE SCREEN
RETURN
* EOP: ShowPick

PROCEDURE Cleanup
*-- test whether report option was selected
DO CASE
CASE gc_dev='CON'
   ? "  Press any key to continue..."
   xx=INKEY(0)
CASE gc_dev='PRN'
   SET PRINT OFF
   SET PRINTER TO
CASE gc_dev='TXT'
   CLOSE ALTERNATE
ENDCASE
gc_dev='CON'
RETURN

* EOP: Cleanup
```

```
PROCEDURE Pause
PARAMETER lc_msg
*-- Parameters : lc_msg = message line
IF TYPE("lc_message")="U"
   gr_error=ERROR()
ENDIF
lc_msg = lc_msg
lc_option='0'
ACTIVATE WINDOW Pause
IF gn_error > 0
   IF TYPE("lc_message")="U"
      @ 0,1 SAY [An error has occurred !! - Error message: ]+MESSAGE()
   ELSE
      @ 0,1 SAY [Error # ]+lc_message
   ENDIF
ENDIF
@ 1,1 SAY lc_msg
WAIT " Press any key to continue..."
DEACTIVATE WINDOW Pause
RETURN

* EOP: Pause

PROCEDURE Multerr
*-- set the global error variable
gn_error=ERROR()
*-- contains error number to test
lc_erno=STR(ERROR(),3)+','
*-- option var.
lc_opt='T'
*-- Dialog box for options Try again and Return to menu.
IF lc_erno $ "108,109,128,129,"
   ACTIVATE WINDOW Pause
   @ 0,2 SAY lc_erno+" "+MESSAGE()
   @ 2,22 SAY "T = Try again, R = Return to menu." GET lc_opt ;
PICTURE "!" VALID lc_opt $ "TR"
   READ
   DEACTIVATE WINDOW Pause
   IF lc_opt = "R"
      RETURN
   ENDIF
ENDIF
*-- Display message and return to menu.
IF .NOT. lc_erno $ "108,109,128,129,"
   DO PAUSE WITH ERROR()
   RETURN
ENDIF
*-- reset global variable
gn_error=0
*-- Try the command again
RETRY
RETURN
```

```
* EOP: Multerr

PROCEDURE Trace
*  Desc: Trace procedure - to let programmer know what module
*           is about to execute and what module has executed.
PARAMETERS p_msg, p_lvl
*-- Parameters : p_msg = message line, p_lvl = trace level
lc_msg = p_msg
ln_lvl = p_lvl
lc_trp = ' '
IF gn_trace < ln_lvl
   RETURN
ENDIF
DEFINE WINDOW trace FROM 5,0 TO 16,79 DOUBLE
ACTIVATE WINDOW trace
DO WHILE lc_trp <> 'Q'
   CLEAR
   @ 2,40-LEN(lc_msg)/2 SAY lc_msg
   @ 4,05 SAY 'S - Set trace level, D - Display status, M - display Memory'
   @ 5,05 SAY 'P - Turn printer on, Q - to Quit'
   lc_trp = 'Q'
   @ 5,38 GET lc_trp PICTURE "!"
   READ
   DO CASE
   CASE lc_trp = 'S'
      @ 2,01 CLEAR
      @ 2,33 SAY 'Set trace level'
      @ 4,05 SAY 'Enter trace level to change to:' GET gn_trace PICTURE '#'
      @ 5,05 SAY '                                '
      READ
      IF gn_trace=0
         @ 2,01 CLEAR
         @ 3,05 SAY 'Trace is now turned off..To reactivate Trace - Press [F3]
         @ 4,05 say 'Press any key to continue...'
         WAIT ''
         ON KEY LABEL F3 gn_trace = 1
      ENDIF
   CASE lc_trp = 'D'
      DISPLAY STATUS
      WAIT
   CASE lc_trp = 'M'
      DISPLAY MEMORY
      WAIT
   CASE lc_trp = 'P'
      SET PRINT ON
   ENDCASE
ENDDO
SET PRINT OFF
RELEASE WINDOW trace
@ 24,79 SAY " "
RETURN
```

```
* EOP: Trace

PROCEDURE PrintSet
*-- Initialize variables
gc_dev='CON'
lc_choice=' '
gn_pkey=0
gn_send=3

DEFINE WINDOW printemp FROM 08,25 TO 17,56

DEFINE POPUP SavePrin FROM 10,40
DEFINE BAR 1 OF SavePrin PROMPT " Send output to ..." SKIP
DEFINE BAR 2 OF SavePrin PROMPT REPLICATE(CHR(196),24) SKIP
DEFINE BAR 3 OF SavePrin PROMPT " CON:   Console" MESSAGE "Send output to Screen"
DEFINE BAR 4 OF SavePrin PROMPT " LPT1:  Parallel port 1 " MESSAGE "Send output to
LPT1:"
DEFINE BAR 5 OF SavePrin PROMPT " LPT2:  Parallel port 2" MESSAGE "Send output to
LPT2:"
DEFINE BAR 6 OF SavePrin PROMPT " COM1:  Serial port 1" MESSAGE "Send output to
COM1:"
DEFINE BAR 7 OF SavePrin PROMPT " FILE = REPORT.TXT" MESSAGE "Send output to
File Report.txt"
ON SELECTION POPUP SavePrin DO get_sele

ACTIVATE POPUP SavePrin
RELEASE POPUP SavePrin

gn_pkey=LASTKEY()
IF gn_send = 7
   gc_dev = 'TXT'
   SET ALTERNATE TO REPORT.TXT
   SET ALTERNATE ON
ELSE
   IF .NOT. (gn_send = 3 .OR. LASTKEY() = 27)
      gc_dev = 'PRN'
      temp = SUBSTR("    LPT1LPT2COM1 ",((gn_send-2)-1)*4,4)
      ON ERROR DO prntrtry
      SET PRINTER TO &temp.
      IF gn_pkey <> 27
         SET PRINT ON
      ENDIF
      ON ERROR
   ENDIF
ENDIF
RELEASE WINDOW printemp
RETURN

PROCEDURE prntrtry
PRIVATE lc_escape
lc_escape = SET("ESCAPE")
```

```
IF .NOT. PRINTSTATUS()
   IF lc_escape = "ON"
      SET ESCAPE OFF
   ENDIF
   gn_pkey = 0
   ACTIVATE WINDOW printemp
   @ 1,0 SAY "Please ready your printer or"
   @ 2,0 SAY "     press ESC to cancel"
   DO WHILE ( .NOT. PRINTSTATUS()) .AND. gn_pkey <> 27
      gn_pkey = INKEY()
   ENDDO
   DEACTIVATE WINDOW printemp
   SET ESCAPE &lc_escape.
   IF gn_pkey <> 27
      RETRY
   ENDIF
ENDIF
RETURN

* EOP: PrintSet

PROCEDURE Position
IF LEN(DBF()) = 0
   DO Pause WITH "Database not in use. "
   RETURN
ENDIF
SET SPACE ON
SET DELIMITERS OFF
ln_type=0           && sublevel selection
ln_rkey=READKEY()   && test for ESC or Return
ln_rec=RECNO()      && DBF record number
ln_num=0            && for input of a number
ld_date=DATE()      && for input of a date
lc_option='0'       && main option ie. Seek, Goto and Locate
*-- Scope ie. ALL, REST, NEXT <n>
STORE SPACE(10) TO lc_scp
*-- 1 = Character SEEK, 2 = For clause, 3 = While clause
STORE SPACE(40) TO lc_ln1, lc_ln2, lc_ln3
lc_temp=""
@ 0,00 SAY "Index order: "+IIF(""=ORDER(),"Database is in natural order",ORDER())
@ 1,00 SAY "Listed below are the first 16 fields."
lc_temp=REPLICATE(CHR(196),19)
@ 2,0 SAY CHR(218)+lc_temp+CHR(194)+lc_temp+CHR(194)+lc_temp+CHR(194)+lc_temp
ln_num=240
DO WHILE ln_num < 560
   lc_temp=FIELD( (ln_num-240)/20 +1)
   @ (ln_num/80),MOD(ln_num,80) SAY CHR(179)+;
lc_temp+SPACE(11-LEN(lc_temp))+;
SUBSTR("= Char  = Date  = Logic = Num   = Float = Memo          ",;
AT(TYPE(lc_temp),"CDLNFMU")*8-7,8)
   ln_num=ln_num+20
ENDDO
```

```
ln_num=1

DEFINE POPUP Posit1 FROM 8,30
DEFINE BAR 1 OF Posit1 PROMPT " Position by " SKIP
DEFINE BAR 2 OF Posit1 PROMPT REPLICATE(CHR(196),15) SKIP
DEFINE BAR 3 OF Posit1 PROMPT " SEEK Record" MESSAGE "Search on index key" SKIP
FOR ""=ORDER()
DEFINE BAR 4 OF Posit1 PROMPT " GOTO Record" MESSAGE "Position to specific record"
DEFINE BAR 5 OF Posit1 PROMPT " LOCATE Record " MESSAGE "Locate record for
condition"
DEFINE BAR 6 OF Posit1 PROMPT " Return" MESSAGE "Return without positioning"
ON SELECTION POPUP Posit1 DO get_sele

SET CONFIRM ON
DO WHILE lc_option='0'
  ACTIVATE POPUP Posit1
  lc_option = ltrim(str(gn_send))  && for popup
   IF LASTKEY() = 27 .OR. lc_option="6"
      GOTO ln_rec
      EXIT
   ENDIF
   DO CASE
   CASE lc_option='3'
      *-- Seek
      IF LEN(NDX(1))=0 .AND. LEN(MDX(1))=0
         DO Pause WITH "Can't use this option - No index files are open."
         LOOP
      ENDIF
      ln_type=1
      lc_ln1=SPACE(40)
      DEFINE WINDOW Posit2 FROM 8,19 TO 15,62 DOUBLE
      ACTIVATE WINDOW Posit2
      @ 1,1 SAY "Enter the type of expression:" GET ln_type PICT "#" RANGE 1,3
      @ 2,1 SAY "(1=character, 2=numeric and 3=date.)"
      SET CURSOR ON
      READ
      SET CURSOR OFF
      IF .NOT. (READKEY() = 12 .OR. READKEY() = 268)
         SET CONFIRM ON
         @ 3,1 SAY "Enter the key expression to search for:"
         IF ln_type=3
            @ 4,1 GET ld_date PICT "@D"
         ELSE
            IF ln_type=2
               @ 4,1 GET ln_num PICT "##########"
            ELSE
               @ 4,1 GET lc_ln1
            ENDIF
         ENDIF
         SET CURSOR ON
         READ
         SET CURSOR OFF
         SET CONFIRM OFF
         IF .NOT. (READKEY() = 12 .OR. READKEY() = 268)
            lc_temp=IIF(ln_type=1,"TRIM(lc_ln1)",IIF(ln_type=2,"ln_num","ld_date"))
            SEEK &lc_temp.
         ENDIF
      ENDIF
      RELEASE WINDOWS Posit2
   CASE lc_option='4'
      *-- Goto
      ln_type=1
      DEFINE POPUP Posit2 FROM 8,30
      DEFINE BAR 1 OF Posit2 PROMPT " GOTO:" SKIP
      DEFINE BAR 2 OF Posit2 PROMPT REPLICATE(CHR(196),10) SKIP
      DEFINE BAR 3 OF Posit2 PROMPT " TOP" MESSAGE "GOTO Top of File"
      DEFINE BAR 4 OF Posit2 PROMPT " BOTTOM" MESSAGE "GOTO Bottom of File"
      DEFINE BAR 5 OF Posit2 PROMPT " Record # " MESSAGE "GOTO A Specific Record"
      ON SELECTION POPUP Posit2 DO get_sele
      ACTIVATE POPUP posit2
      ln_type = gn_send
      IF LASTKEY() <> 27
         IF ln_type=5
            DEFINE WINDOW Posit2 FROM 8,26 TO 13,50 DOUBLE
            ACTIVATE WINDOW Posit2
            ln_num=0
            @ 3,1 SAY "Max. Record # = "+LTRIM(STR(RECCOUNT()))
            @ 1,1 SAY "Record to GOTO" GET ln_num PICT "######" RANGE 1,RECCOUNT()
            SET CURSOR ON
            READ
            SET CURSOR OFF
            IF .NOT. (READKEY() = 12 .OR. READKEY() = 268)
               GOTO ln_num
            ENDIF
            RELEASE WINDOWS Posit2
         ELSE
           lc_temp=IIF(ln_type=3,"TOP","BOTTOM")
           GOTO &lc_temp.
         ENDIF
      ENDIF
   CASE lc_option='5'
      *-- Locate
      DEFINE WINDOW Posit2 FROM 8,16 TO 14,66 DOUBLE
      ACTIVATE WINDOW Posit2
      @ 1,19 SAY "ie. ALL, NEXT <n>, and REST"
      @ 1,01 SAY "Scope:" GET lc_scp
      @ 2,01 SAY "For:  " GET lc_ln2
      @ 3,01 SAY "While:" GET lc_ln3
      SET CURSOR ON
      READ
      SET CURSOR OFF
      IF .NOT. (READKEY() = 12 .OR. READKEY() = 268)
         lc_temp=TRIM(lc_scp)
         lc_temp=lc_temp + IIF(LEN(TRIM(lc_ln2)) > 0," FOR "+TRIM(lc_ln2),"")
         lc_temp=lc_temp + IIF(LEN(TRIM(lc_lnD)) > 0," WHILE "+TRIM(lc_lnD),"")
         IF LEN(lc_temp) > 0
            LOCATE &lc_temp.
         ELSE
            DO Pause WITH 'All fields were blank.'
         ENDIF
      ENDIF
      RELEASE WINDOW Posit2
   ENDCASE
   IF EOF()
      DO Pause WITH "Record not found."
      GOTO ln_rec
   ENDIF
   IF READKEY()=12 .OR. READKEY()= 268 .OR. LASTKEY()=27  && Esc was hit
      lc_option='0'
   ENDIF
ENDDO
SET CURSOR &gc_cursor.
SET DELIMITERS &gc_deli.
SET CONFIRM OFF
RETURN

* EOP: Position

PROCEDURE BefAct
SAVE SCREEN TO Browscr&lc_ApGen.
DEACTIVATE WINDOW Fullscr
SET SCOREBOARD ON
RETURN
* EOP: BefAct

PROCEDURE AftAct
CLEAR
SET SCOREBOARD OFF
ACTIVATE WINDOW Fullscr
RESTORE SCREEN FROM Browscr&lc_ApGen.
RELEASE SCREEN Browscr&lc_ApGen.
RETURN
* EOP: AftAct

PROCEDURE Postnhlp
ln_getkey=INKEY()
DEFINE WINDOW Temphelp FROM 3,12 TO 19,67
ACTIVATE WINDOW Temphelp
DO CASE
CASE "SEEK" $ PROMPT()
*-- HELP SEEK
? " SEEK <exp>"
?
? " Evaluates a specified expression and attempts to"
? " find its value in the master index of the database"
? " file.  Returns a logical true (.T.) if the index"
? " key is found, and a logical false (.F.) if it is"
? " not found."
?
? " Ex: SEEK CTOD('11/03/87')  -  converts the"
? "     expression from character to date and"
? "     then searches for the value in the index"
?
CASE LEFT(LTRIM(PROMPT()),4) $ "GOTO TOP BOTT Reco"
*-- HELP GOTO
? " GO/GOTO BOTTOM/TOP [IN <alias>]"
? " or"
? " GO/GOTO [RECORD] <record number> [IN <alias>]"
? " or"
? " <record number>"
?
? " Positions the record pointer to a specified record"
? " or location in the active database file."
?
? "      TOP moves the pointer to the first record"
? "      BOTTOM moves the pointer to the last record"
?
? " Ex: 4  -  moves the record pointer to record 4"
?
CASE "LOCATE" $ PROMPT()
*-- HELP LOCATE
? " LOCATE FOR <condition> [<scope>]"
? "      [WHILE <condition>]"
?
? " Searches the active database file, sequentially,"
? " for the first record that meets the specified"
? " criteria.  The function FOUND() returns true (.T.)"
? " if LOCATE is successful."
?
? " Ex: LOCATE FOR Age = '25' NEXT 5"
? "     searches for the next five records"
? "     containing 25 in the Age field"
?
CASE "Return" $ PROMPT()
?
? " Return to action in progress, with or without"
? " positioning the record pointer."
ENDCASE
ln_getkey = INKEY(0)
DEACTIVATE WINDOW Temphelp
RELEASE WINDOW Temphelp
RETURN
* EOP: Postnhlp

FUNCTION Color
*--------------------------------------------------------------------------
* Format:
* COLOR( <expC> )
*  <expC> = NORMAL, HIGHLIGHT, MESSAGES, TITLES, BOX, INFORMATION,
FIELDS
*        or a variable with all colors store in it
*  Ver: dBASE 1.1
*
* The COLOR() function either returns or sets colors returned with the
* SET("attribute") setting
* If <expC> is a color string then null is returned otherwise the color
* setting is returned for one of dBASE's color options
*
* See Also: SET("attribute")
*
*-------------------------------------------------------------------------
PARAMETERS set_color
PRIVATE color_num, color_str, cnt

set_color = UPPER(set_color)
IF set_color = "COLOR"
  *- Return standard, enhanced, border colors only
  RETURN SUBSTR(SET("attr"),1, AT(" &", SET("attr")))
ENDIF

*- Declare array to parse color options from SET("attr")
PRIVATE color_
DECLARE color_[8]
*- Determine if user is restoring colors vs. saving colors
IF " &" $ set_color
  color_str = ","+set_color+","                   && Restore color attributes
ELSE
  color_str = ","+SET("ATTRIBUTE")+","            && Save color attributes
ENDIF

*-- Stuff array with individual color setting
color_str = STUFF(color_str, AT(" &", color_str), 4, ",")
cnt = 1
DO WHILE cnt <= 8
  color_str = SUBSTR(color_str, AT(",", color_str ) +1 )
  color_[cnt] = SUBSTR(color_str, 1, AT(",", color_str ) - 1)
  cnt = cnt + 1
ENDDO

IF " &" $ set_color
  *-- Set color back
  SET COLOR TO ,,&color_[3].                      && Border color
  SET COLOR OF NORMAL TO &color_[1].
  SET COLOR OF HIGHLIGHT TO &color_[2].
  SET COLOR OF MESSAGES TO &color_[4].
  SET COLOR OF TITLES TO &color_[5].
  SET COLOR OF BOX TO &color_[6].
  SET COLOR OF INFORMATION TO &color_[7].
  SET COLOR OF FIELDS TO &color_[8].
ELSE
```

```
  *-- Return color string requested
  DO CASE
  CASE set_color $ "NORMAL"
    color_num = 1
  CASE set_color $ "HIGHLIGHT"
    color_num = 2
  CASE set_color $ "BORDER"
    color_num = 3
  CASE set_color $ "MESSAGES"
    color_num = 4
  CASE set_color $ "TITLES"
    color_num = 5
  CASE set_color $ "BOX"
    color_num = 6
  CASE set_color $ "INFORMATION"
    color_num = 7
  CASE set_color $ "FIELDS"
    color_num = 8
  ENDCASE
ENDIF
RETURN IIF(" &" $ set_color, "", color_[color_num])



******************************************************************************
* Program......: MPDEF
* Author.......:          This is an APPLICATION OBJECT.
* Date.........: 8-04-91
* Notice.......: Type information here or greetings to your users.
* dBASE Ver....: See Application menu to use as sign-on banner.
* Generated by.: APGEN version 1.3
* Description..: user application of dive log database.

* Description..: Defines all menus in the system
******************************************************************************
PROCEDURE MPDEF


****************
PARAMETER BATON
IF .NOT. BATON="GOOD"
  DO VIOLATIO
ELSE
****************

IF gl_color
   SET COLOR OF NORMAL TO W+/B
   SET COLOR OF MESSAGES TO W+/N
   SET COLOR OF TITLES TO W/B
   SET COLOR OF HIGHLIGHT TO RG+/GB
   SET COLOR OF BOX TO RG+/GB
   SET COLOR OF INFORMATION TO B/W
   SET COLOR OF FIELDS TO N/GB
ENDIF
CLEAR

*-- Sign-on banner

SET BORDER TO
@ 5,9 TO 16,69 DOUBLE COLOR RG+/GB
@ 7,10 SAY "   * * * WELCOME TO AUTOMATED DIVE LOG VERSION 1.1 * * *"
@ 9,10 SAY "  This user application allows for entering dives, find-"
@ 10,10 SAY "  ing dive logs for printing or browzing, and finding and"
@ 11,10 SAY "  printing of qualification lists.  Security of informa-"
@ 12,10 SAY "  tion is ensured if users keep their diver number secure."
@ 13,10 SAY "  You must know your diver number to accomplish any of the"
@ 14,10 SAY "  systems functions.  Thank you for using ADL 1.1!"
@ 6,10 FILL TO 15,68 COLOR W+/N
@ 24,30 SAY " Press "+CHR(17)+CHR(196)+CHR(217)+" to continue. "
gn_ikey=INKEY(500)

CLEAR

*-- Prevents clearing of menus from commands:
*-- SET STATUS and SET SCOREBOARD
DEFINE WINDOW FullScr FROM 0,0 TO 24,79 NONE
*-- Position at runtime and batch process
DEFINE WINDOW Savescr FROM 0,0 TO 21,79 NONE
*-- F1 Help
DEFINE WINDOW Helpscr FROM 0,0 TO 21,79 NONE
IF gn_ApGen=1
   *-- Are you sure? (exit application)
   DEFINE WINDOW Exit_App FROM 11,17 TO 15,62 DOUBLE
   *-- Pause message box
   DEFINE WINDOW Pause FROM 15,00 TO 19,79 DOUBLE
ENDIF

ACTIVATE WINDOW FullScr
@ 24,00
@ 23,00 SAY "Loading..."
SET BORDER TO DOUBLE
*-- Bar
DEFINE MENU USERBAR MESSAGE "Select an option with the arrow keys and push
ENTER."
DEFINE PAD PAD_1 OF USERBAR PROMPT "Dive" AT 1,1
ON SELECTION PAD PAD_1 OF USERBAR DO ACT01
DEFINE PAD PAD_2 OF USERBAR PROMPT "Log" AT 1,10
ON SELECTION PAD PAD_2 OF USERBAR DO ACT01
DEFINE PAD PAD_3 OF USERBAR PROMPT "Quals" AT 1,18
ON SELECTION PAD PAD_3 OF USERBAR DO ACT01
DEFINE PAD PAD_4 OF USERBAR PROMPT "Exit" AT 1,28
ON SELECTION PAD PAD_4 OF USERBAR DO ACT01
?? ","
SET BORDER TO DOUBLE
```

```
*-- Popup
DEFINE POPUP DIVE FROM 2,1 TO 4,17 ;
MESSAGE "Press ENTER to continue/enter a dive or else --> (arrow key)"
DEFINE BAR 1 OF DIVE PROMPT "Enter a dive"
ON SELECTION POPUP DIVE DO ACT02
?? "."
SET BORDER TO DOUBLE
*-- Popup
DEFINE POPUP LOG FROM 2,10 TO 6,29 ;
MESSAGE "Choose an option with arrow keys and push RETURN or else use --> (arrow key)"
DEFINE BAR 1 OF LOG PROMPT "Find a log"
DEFINE BAR 2 OF LOG PROMPT "Browse found log"
DEFINE BAR 3 OF LOG PROMPT "Print found log"
************
ON SELECTION POPUP LOG DO ACT03 WITH "GOOD"
*************
?? "."
SET BORDER TO DOUBLE
*-- Popup
DEFINE POPUP QUAL FROM 2,18 TO 5,42 ;
MESSAGE "Use arrow keys to select an option and press RETURN or else --> (arrow keys)"
DEFINE BAR 1 OF QUAL PROMPT "Find a qual list"
DEFINE BAR 2 OF QUAL PROMPT "Print found qual list"
************
ON SELECTION POPUP QUAL DO ACT04 WITH "GOOD"
************
?? "."
SET BORDER TO DOUBLE
*-- Popup
DEFINE POPUP EXIT FROM 2,28 TO 4,49 ;
MESSAGE 'Position: '+CHR(27)+CHR(26)+CHR(25)+CHR(24)+'  Select:
'+CHR(17)+CHR(196)+CHR(217)+'  Help: F1'
DEFINE BAR 1 OF EXIT PROMPT "Return to DBASE IV"
ON SELECTION POPUP EXIT DO ACT05
?? "."
****************
ENDIF
****************

RETURN
*-- EOP: MPDEF.PRS

PROCEDURE 1HELP1
ln_key=INKEY()
ON KEY LABEL F1
lc_popmenu=IIF( "" = POPUP(), MENU(), POPUP() )
ACTIVATE WINDOW Helpscr
SET ESCAPE OFF
ACTIVATE SCREEN
@ 0,0 CLEAR TO 21,79
@ 1,0 TO 21,79 COLOR RG+/GB
@ 24,00
@ 24,26 SAY "Press any key to continue..."
@ 0,0 SAY ""
DO CASE
*-- help for menu USERBAR
CASE "USERBAR" = lc_popmenu
   @ 2,2 SAY "No Help defined."
*-- help for menu DIVE
CASE "DIVE" = lc_popmenu
   @ 2,2 SAY "No Help defined."
*-- help for menu LOG
CASE "LOG" = lc_popmenu
   @ 2,2 SAY "No Help defined."
*-- help for menu QUAL
CASE "QUAL" = lc_popmenu
   @ 2,2 SAY "No Help defined."
*-- help for menu EXIT
CASE "EXIT" = lc_popmenu
   @ 2,2 SAY "No Help defined."
OTHERWISE
   @ 2,2 SAY "Unknown menu name, help was never defined."
ENDCASE
ln_key=INKEY(0)
SET ESCAPE ON
@ 24,00
DEACTIVATE WINDOW Helpscr
ON KEY LABEL F1 DO 1HELP1
RETURN
*-- EOP: 1HELP1




**************************************************************************
* Program......: USERBAR.PRG
* Author.......:          This is an APPLICATION OBJECT.
* Date.........: 8-04-91
* Notice.......: Type information here or greetings to your users.
* dBASE Ver....: See Application menu to use as sign-on banner.
* Generated by.: APGEN version 1.3
* Description..: FIRST MENU LEVEL IN USER APPLICATION.

* Description..: Menu actions
**************************************************************************
PROCEDURE USERBAR
PARAMETERS entryflg, BATON
PRIVATE gc_prognum
gc_prognum="01"
SET COLOR OF NORMAL TO W+/B
CLEAR
PRIVATE lc_ApGen
lc_ApGen=LTRIM(STR(gn_ApGen))
```

```
DO SET01
IF gn_error > 0
   gn_error=0
   RETURN
ENDIF

*-- Before menu code


ACTIVATE MENU USERBAR

@ 0,0 CLEAR TO 2,79

*-- After menu

RETURN
*-- EOP USERBAR

PROCEDURE SET01
ON KEY LABEL F1 DO 1HELP1

DO DBF01 && open menu level database

IF gn_error = 0
   IF gl_color .AND. .NOT. SET("ATTRIBUTE") = "W+/B,RG+/GB,N/N "+;
      CHR(38)+CHR(38)+" W+/N,W/B,RG+/GB,B/W,N/GB"
      SET COLOR OF NORMAL TO W+/B
      SET COLOR OF MESSAGES TO W+/N
      SET COLOR OF TITLES TO W/B
      SET COLOR OF HIGHLIGHT TO RG+/GB
      SET COLOR OF BOX TO RG+/GB
      SET COLOR OF INFORMATION TO B/W
      SET COLOR OF FIELDS TO N/GB
   ENDIF

   SET BORDER TO
   @ 0,0 TO 2,79 DOUBLE COLOR RG+/GB
   @ 1,1 CLEAR TO 1,78
   @ 1,1 FILL TO 1,78 COLOR W+/N
   @ 1,1 SAY "Dive" COLOR W+/N
   @ 1,10 SAY "Log" COLOR W+/N
   @ 1,18 SAY "Quals" COLOR W+/N
   @ 1,28 SAY "Exit" COLOR W+/N
ENDIF
RETURN

PROCEDURE DBF01
CLOSE DATABASES
*-- Open menu level view/database
lc_message="0"
ON ERROR lc_message=LTRIM(STR(ERROR()))+" "+MESSAGE()
USE DIVE.DBF
```

```
ON ERROR
gn_error=VAL(lc_message)
IF gn_error > 0
   DO Pause WITH ;
   "Error opening DIVE.DBF"
   lc_new='Y'
   RETURN
ENDIF
lc_new='Y'
RELEASE lc_message
RETURN

PROCEDURE ACT01
*-- Begin USERBAR: BAR Menu Actions.
*-- (before item, action, and after item)
*
PRIVATE lc_new, lc_dbf
lc_new=' '
lc_dbf=' '
DO CASE
CASE "PAD_1" = PAD()
   lc_new='Y'
   DO DIVE WITH " 01"
CASE "PAD_2" = PAD()
   lc_new='Y'
   DO LOG WITH " 01"
CASE "PAD_3" = PAD()
   lc_new='Y'
   DO QUAL WITH " 01"
CASE "PAD_4" = PAD()
   lc_new='Y'
   DO EXIT WITH " 01"
OTHERWISE
   @ 24,00
   @ 24,21 SAY "This item has no action. Press a key."
   x=INKEY(0)
   @ 24,00
ENDCASE
SET MESSAGE TO
IF gc_quit='Q'
   IF LEFT(entryflg,1) = "B"
      DEACTIVATE MENU
   ELSE
      DEACTIVATE MENU && USERBAR
   ENDIF
ENDIF
IF lc_new='Y'
   lc_file="SET"+gc_prognum
   DO &lc_file.
ENDIF
RETURN
```

```
******************************************************************************
* Program......: DIVE.PRG
* Author.......:         This is an APPLICATION OBJECT.
* Date.........: 8-04-91
* Notice.......: Type information here or greetings to your users.
* dBASE Ver....: See Application menu to use as sign-on banner.
* Generated by.: APGEN version 1.3
* Description..: User popup for entering a dive.

* Description..: Menu actions
******************************************************************************
PROCEDURE DIVE
PARAMETER entryflg
PRIVATE gc_prognum
gc_prognum="02"

DO SET02
IF gn_error > 0
  gn_error=0
  RETURN
ENDIF

*-- Before menu code


ACTIVATE POPUP DIVE

*-- After menu

RETURN
*-- EOP DIVE

PROCEDURE SET02
ON KEY LABEL F1 DO 1HELP1

DO DBF02 && open menu level database

IF gn_error = 0
  IF gl_color .AND. .NOT. SET("ATTRIBUTE") = "W+/B,RG+/GB,N/N "+;
    CHR(38)+CHR(38)+" W+/N,W/B,RG+/GB,B/W,N/GB"
    SET COLOR OF NORMAL TO W+/B
    SET COLOR OF MESSAGES TO W+/N
    SET COLOR OF TITLES TO W/B
    SET COLOR OF HIGHLIGHT TO RG+/GB
    SET COLOR OF BOX TO RG+/GB
    SET COLOR OF INFORMATION TO B/W
    SET COLOR OF FIELDS TO N/GB
  ENDIF
ENDIF
RETURN
```

```
PROCEDURE DBF02
CLOSE DATABASES
*-- Open menu level view/database
lc_message="0"
ON ERROR lc_message=LTRIM(STR(ERROR()))+" "+MESSAGE()
USE DIVE.DBF
ON ERROR
gn_error=VAL(lc_message)
IF gn_error > 0
   DO Pause WITH ;
   "Error opening DIVE.DBF"
   lc_new='Y'
   RETURN
ENDIF
lc_new='Y'
RELEASE lc_message
RETURN

PROCEDURE ACT02
*-- Begin DIVE: POPUP Menu Actions.
*-- (before item, action, and after item)
*
PRIVATE lc_new, lc_dbf
lc_new=' '
lc_dbf=' '
DO CASE
CASE BAR() = 1
   IF .NOT. gl_batch
      DO BefAct
   ENDIF
   SET SCOREBOARD ON
   SET MESSAGE TO
   *-- Desc: attach format file DIVEFORM
   SET FORMAT TO DIVEFORM
   APPEND

   *-- close format file so as not to affect READ's
   SET FORMAT TO
   SET SCOREBOARD OFF
   IF .NOT. gl_batch
      DO AftAct
   ENDIF
 ENDCASE
 SET MESSAGE TO
 IF gc_quit='Q'
    IF LEFT(entryflg,1) = "B"
       DEACTIVATE MENU
    ELSE
       DEACTIVATE POPUP && DIVE
    ENDIF
 ENDIF
 IF lc_new='Y'
```

```
     lc_file="SET"+gc_prognum
     DO &lc_file.
  ENDIF
  RETURN
  ******************************************************************************
  * Program......: LOG.PRG
  * Author.......:          This is an APPLICATION OBJECT.
  * Date.........: 8-04-91
  * Notice.......: Type information here or greetings to your users.
  * dBASE Ver....: See Application menu to use as sign-on banner.
  * Generated by.: APGEN version 1.3
  * Description..: Log popup in user application of dive log

  * Description..: Menu actions
  ******************************************************************************
  PROCEDURE LOG
  PARAMETER entryflg
  PRIVATE gc_prognum
  gc_prognum="03"

  DO SET03
  IF gn_error > 0
     gn_error=0
     RETURN
  ENDIF

  *-- Before menu code


  ACTIVATE POPUP LOG

  *-- After menu

  RETURN
  *-- EOP LOG

  PROCEDURE SET03
  ON KEY LABEL F1 DO 1HELP1

  DO DBF03 && open menu level database

  IF gn_error = 0
     IF gl_color .AND. .NOT. SET("ATTRIBUTE") = "W+/B,RG+/GB,N/N "+;
        CHR(38)+CHR(38)+" W+/N,W/B,RG+/GB,B/W,N/GB"
        SET COLOR OF NORMAL TO W+/B
        SET COLOR OF MESSAGES TO W+/N
        SET COLOR OF TITLES TO W/B
        SET COLOR OF HIGHLIGHT TO RG+/GB
        SET COLOR OF BOX TO RG+/GB
        SET COLOR OF INFORMATION TO B/W
        SET COLOR OF FIELDS TO N/GB
    ENDIF
```

```
ENDIF
RETURN

PROCEDURE DBF03
CLOSE DATABASES
*-- Open menu level view/database
lc_message="0"
ON ERROR lc_message=LTRIM(STR(ERROR()))+" "+MESSAGE()
USE DIVE.DBF
ON ERROR
gn_error=VAL(lc_message)
IF gn_error > 0
   DO Pause WITH ;
   "Error opening DIVE.DBF"
   lc_new='Y'
   RETURN
ENDIF
lc_new='Y'
RELEASE lc_message
RETURN

PROCEDURE ACT03
*-- Begin LOG: POPUP Menu Actions.
*-- (before item, action, and after item)
*
*******************

PARAMETER BATON
IF .NOT. BATON='GOOD'
  DO VIOLATIO
ELSE
*********************

PRIVATE lc_new, lc_dbf
lc_new=' '
lc_dbf=' '
DO CASE
CASE BAR() = 1
   IF .NOT. gl_batch
      DO BefAct
   ENDIF
   SET SCOREBOARD ON
   SET MESSAGE TO
*******************************
   DO JOIN1.PRG WITH "GOOD"
********************************
   SET SCOREBOARD OFF
   IF .NOT. gl_batch
      DO AftAct
   ENDIF
CASE BAR() = 2
   *-- Open Item level view/database and indexes
```

```
    CLOSE DATABASES
    lc_dbf='Y'
    lc_message="0"
    ON ERROR lc_message=LTRIM(STR(ERROR()))+" "+MESSAGE()
    USE TEMP.DBF
    ON ERROR
    gn_error=VAL(lc_message)
    IF gn_error > 0
       DO Pause WITH ;
       "Error opening TEMP.DBF"
       gn_error=0
       lc_file="SET"+gc_prognum
       DO &lc_file.
       RETURN
    ENDIF
    lc_new='Y'
    RELEASE lc_message
    IF .NOT. gl_batch
       DO BefAct
    ENDIF
    SET MESSAGE TO
    *-- Desc: Report
    REPORT FORM LOG_REPO PLAIN
    WAIT
    IF .NOT. gl_batch
       DO AftAct
    ENDIF
  CASE BAR() = 3
    *-- Open Item level view/database and indexes
    CLOSE DATABASES
    lc_dbf='Y'
    lc_message="0"
    ON ERROR lc_message=LTRIM(STR(ERROR()))+" "+MESSAGE()
    USE TEMP.DBF
    ON ERROR
    gn_error=VAL(lc_message)
    IF gn_error > 0
       DO Pause WITH ;
       "Error opening TEMP.DBF"
       gn_error=0
       lc_file="SET"+gc_prognum
       DO &lc_file.
       RETURN
    ENDIF
    lc_new='Y'
    RELEASE lc_message
    IF .NOT. gl_batch
       DO BefAct
    ENDIF
    SET MESSAGE TO
    *-- Desc: Report
    SET PRINT ON
```

```
   REPORT FORM LOG_REPO PLAIN  NOEJECT
   SET PRINT OFF
   IF .NOT. gl_batch
      DO AftAct
   ENDIF
ENDCASE
SET MESSAGE TO
IF gc_quit='Q'
   IF LEFT(entryflg,1) = "B"
      DEACTIVATE MENU
   ELSE
      DEACTIVATE POPUP && LOG
   ENDIF
ENDIF
IF lc_new='Y'
   lc_file="SET"+gc_prognum
   DO &lc_file.
ENDIF
IF lc_dbf='Y' .AND. .NOT. lc_new='Y'
   lc_file="DBF"+gc_prognum
   DO &lc_file.
ENDIF
****************
ENDIF
******************



RETURN
******************************************************************************
* Program......: QUAL.PRG
* Author.......:          This is an APPLICATION OBJECT.
* Date.........: 8-04-91
* Notice.......: Type information here or greetings to your users.
* dBASE Ver....: See Application menu to use as sign-on banner.
* Generated by.: APGEN version 1.3
* Description..: qual popup for user application for dive log.

* Description..: Menu actions
******************************************************************************
PROCEDURE QUAL
PARAMETER entryflg
PRIVATE gc_prognum
gc_prognum="04"

DO SET04
IF gn_error > 0
   gn_error=0
   RETURN
ENDIF

*-- Before menu code
```

```
ACTIVATE POPUP QUAL

*-- After menu

RETURN
*-- EOP QUAL

PROCEDURE SET04
ON KEY LABEL F1 DO 1HELP1

DO DBF04 && open menu level database

IF gn_error = 0
   IF gl_color .AND. .NOT. SET("ATTRIBUTE") = "W+/B,RG+/GB,N/N "+;
      CHR(38)+CHR(38)+" W+/N,W/B,RG+/GB,B/W,N/GB"
      SET COLOR OF NORMAL TO W+/B
      SET COLOR OF MESSAGES TO W+/N
      SET COLOR OF TITLES TO W/B
      SET COLOR OF HIGHLIGHT TO RG+/GB
      SET COLOR OF BOX TO RG+/GB
      SET COLOR OF INFORMATION TO B/W
      SET COLOR OF FIELDS TO N/GB
   ENDIF
ENDIF
RETURN

PROCEDURE DBF04
CLOSE DATABASES
*-- Open menu level view/database
lc_message="0"
ON ERROR lc_message=LTRIM(STR(ERROR()))+" "+MESSAGE()
USE DIVER.DBF
ON ERROR
gn_error=VAL(lc_message)
IF gn_error > 0
   DO Pause WITH ;
   "Error opening DIVER.DBF"
   lc_new='Y'
   RETURN
ENDIF
lc_new='Y'
RELEASE lc_message
RETURN

PROCEDURE ACT04
*-- Begin QUAL: POPUP Menu Actions.
*-- (before item, action, and after item)
*
***************
PARAMETER BATON
IF .NOT. BATON="GOOD"
  DO VIOLATIO
ELSE
```

```
*************

PRIVATE lc_new, lc_dbf
lc_new=' '
lc_dbf=' '
DO CASE
CASE BAR() = 1
   IF .NOT. gl_batch
      DO BefAct
   ENDIF
   SET SCOREBOARD ON
   SET MESSAGE TO
********************
   DO QUALLIST.PRG WITH "GOOD"
********************
   SET SCOREBOARD OFF
   IF .NOT. gl_batch
      DO AftAct
   ENDIF
CASE BAR() = 2
   *-- Open Item level view/database and indexes
   CLOSE DATABASES
   lc_dbf='Y'
   lc_message="0"
   ON ERROR lc_message=LTRIM(STR(ERROR()))+" "+MESSAGE()
   USE TEMP5.DBF
   ON ERROR
   gn_error=VAL(lc_message)
   IF gn_error > 0
      DO Pause WITH ;
      "Error opening TEMP5.DBF"
      gn_error=0
      lc_file="SET"+gc_prognum
      DO &lc_file.
      RETURN
   ENDIF
   lc_new='Y'
   RELEASE lc_message
   IF .NOT. gl_batch
      DO BefAct
   ENDIF
   SET MESSAGE TO
   *-- Desc: Report
   SET PRINT ON
   REPORT FORM QUALRPT PLAIN  NOEJECT
   SET PRINT OFF
   IF .NOT. gl_batch
      DO AftAct
   ENDIF
ENDCASE
```

```
SET MESSAGE TO
IF gc_quit='Q'
   IF LEFT(entryflg,1) = "B"
      DEACTIVATE MENU
   ELSE
      DEACTIVATE POPUP && QUAL
   ENDIF
ENDIF
IF lc_new='Y'
   lc_file="SET"+gc_prognum
   DO &lc_file.
ENDIF
IF lc_dbf='Y' .AND. .NOT. lc_new='Y'
   lc_file="DBF"+gc_prognum
   DO &lc_file.
ENDIF
*********
ENDIF
*********
RETURN
************************************************************************
* Program......: EXIT.PRG
* Author.......:          This is an APPLICATION OBJECT.
* Date.........: 8-04-91
* Notice.......: Type information here or greetings to your users.
* dBASE Ver....: See Application menu to use as sign-on banner.
* Generated by.: APGEN version 1.3
* Description..:

* Description..: Menu actions
************************************************************************
PROCEDURE EXIT
PARAMETER entryflg
PRIVATE gc_prognum
gc_prognum="05"

DO SET05
IF gn_error > 0
   gn_error=0
   RETURN
ENDIF

*-- Before menu code


ACTIVATE POPUP EXIT

*-- After menu

RETURN
*-- EOP EXIT
```

```
PROCEDURE SET05
ON KEY LABEL F1 DO 1HELP1

DO DBF05 && open menu level database

IF gn_error = 0
   IF gl_color .AND. .NOT. SET("ATTRIBUTE") = "W+/B,RG+/GB,N/N "+;
      CHR(38)+CHR(38)+" W+/N,W/B,RG+/GB,B/W,N/GB"
      SET COLOR OF NORMAL TO W+/B
      SET COLOR OF MESSAGES TO W+/N
      SET COLOR OF TITLES TO W/B
      SET COLOR OF HIGHLIGHT TO RG+/GB
      SET COLOR OF BOX TO RG+/GB
      SET COLOR OF INFORMATION TO B/W
      SET COLOR OF FIELDS TO N/GB
   ENDIF
ENDIF
RETURN

PROCEDURE DBF05
CLOSE DATABASES
*-- Open menu level view/database
lc_message="0"
ON ERROR lc_message=LTRIM(STR(ERROR()))+" "+MESSAGE()
USE DIVE.DBF
ON ERROR
gn_error=VAL(lc_message)
IF gn_error > 0
   DO Pause WITH ;
   "Error opening DIVE.DBF"
   lc_new='Y'
   RETURN
ENDIF
lc_new='Y'
RELEASE lc_message
RETURN

PROCEDURE ACT05
*-- Begin EXIT: POPUP Menu Actions.
*-- (before item, action, and after item)
*
PRIVATE lc_new, lc_dbf
lc_new=' '
lc_dbf=' '
***************
RUN pkzip adldata -m -sIRAQ *.dbf
***************
DO CASE
CASE BAR() = 1
   *-- Return to caller
   gc_quit='Q'
   IF LEFT(entryflg,1) <> "B"
```

```
      DEACTIVATE POPUP && EXIT
   ELSE
      DEACTIVATE MENU
   ENDIF
   RETURN
ENDCASE
SET MESSAGE TO
IF gc_quit='Q'
   IF LEFT(entryflg,1) = "B"
      DEACTIVATE MENU
   ELSE
      DEACTIVATE POPUP && EXIT
   ENDIF
ENDIF
IF lc_new='Y'
   lc_file="SET"+gc_prognum
   DO &lc_file.
ENDIF
RETURN




*******************************************************************************
***
*-- Name.......: DIVEFORM.FMT
*-- Date.......: 8-04-91
*-- Version....: dBASE IV, Format 1.1
*-- Notes......: Format files use "" as delimiters!
*******************************************************************************
***

*-- Format file initialization code -------------------------------------------

*-- Some of these PRIVATE variables are created based on CodeGen and may not
*-- be used by your particular .fmt file
PRIVATE lc_talk, lc_cursor, lc_display, lc_status, lc_carry, lc_proc,;
        ln_typeahd, gc_cut

IF SET("TALK") = "ON"
   SET TALK OFF
   lc_talk = "ON"
ELSE
   lc_talk = "OFF"
ENDIF
lc_cursor = SET("CURSOR")
SET CURSOR ON

lc_status = SET("STATUS")
*-- SET STATUS was ON when you went into the Forms Designer.
IF lc_status = "OFF"
   SET STATUS ON
ENDIF
```

```
*-- @ SAY GETS Processing. --------------------------------------------------

*--  Format Page: 1
@ 0,3 TO 6,56 DOUBLE
@ 2,6 SAY "Dive data entry form"
@ 4,6 SAY "fill in the following data concerning your dive"
@ 8,7 TO 19,50
@ 9,8 SAY "diver I.D. number (SSN):"
@ 9,38 GET Diver_num PICTURE "999-99-9999"
@ 10,8 SAY "date of dive:"
@ 10,38 GET Date
@ 11,8 SAY "serial (nth dive of the day):"
@ 11,38 GET Serial PICTURE "9"
@ 12,8 SAY "day or night:"
@ 12,38 GET Nite_day PICTURE "XXXXX"
@ 12,44 SAY "dive"
@ 13,8 SAY "fresh or salt:"
@ 13,38 GET Fresh_salt PICTURE "XXXXX"
@ 13,44 SAY "water"
@ 14,8 SAY "water temerature:"
@ 14,38 GET Temperatur PICTURE "99"
@ 14,41 SAY "degrees F"
@ 15,8 SAY "maximum depth of dive:"
@ 15,38 GET Depth PICTURE "999"
@ 15,42 SAY "feet"
@ 16,8 SAY "average u/w visibility:"
@ 16,38 GET Visibility PICTURE "999"
@ 16,42 SAY "feet"
@ 17,8 SAY "amount of air consumed:"
@ 17,38 GET Air_used PICTURE "9999"
@ 17,43 SAY "psi"
@ 18,8 SAY "total dive time:"
@ 18,38 GET Time PICTURE "9.9"
@ 18,42 SAY "hours"
READ

*-- Format Page: 2

@ 0,2 SAY "Does the site you dived at have a system site number?  (check on current site"
@ 1,4 SAY "printout)  If it does, enter it here:"
@ 1,42 GET Site_num PICTURE "999"
@ 1.47 SAY "If not, describe the site in the"
@ 2,1 SAY "next data field, dive remarks."
@ 4,2 SAY "dive remarks:"
@ 5,9 GET Dive_rmks PICTURE "@S68
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
@ 6,9 GET Dive_rmks PICTURE "@S68
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

```
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX'
@ 7,9 GET Dive_rmks PICTURE "@S6B
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
*-- Format file exit code ------------------------------------------------------

*-- SET STATUS was ON when you went into the Forms Designer.
IF lc_status = "OFF"  && Entered form with status off
   SET STATUS OFF     && Turn STATUS "OFF" on the way out
ENDIF
SET CURSOR &lc_cursor.
SET TALK &lc_talk.

RELEASE lc_talk,lc_fields,lc_status
*-- EOP: DIVEFORM.FMT




* dBASE IV .PRG file
* JOIN1.PRG (joins dive.dbf with diver.dbf and site.dbf and stores to temp.dbf
* for one diver. fields are set for a dive log.)
**************
PARAMETER BATON
ON ERROR CANCEL
IF .NOT. BATON = "GOOD"
  DO VIOLATIO
ELSE
***************


ACCEPT "Enter the diver number of the diver whos log you wish to find: ";
to number

SET FIELDS TO
SELECT 1
USE DIVE.DBF AGAIN NOUPDATE
USE DIVER.DBF AGAIN NOUPDATE IN 2 ORDER DIVER_NUM
USE SITE.DBF AGAIN NOUPDATE IN 3 ORDER SITE_NUM
SET EXACT ON
SET FILTER TO ((A->DIVER_NUM=number)) .AND. FOUND(2) .AND. FOUND(;
3)
SET RELATION TO A->DIVER_NUM INTO B
SELECT 2
SET RELATION TO A->SITE_NUM INTO C
SELECT 1
SET SKIP TO B,C
GO TOP
SET FIELDS TO A->DATE,A->SERIAL,A->SITE_NUM,A->DIVER_NUM,A-
>FRESH_SALT;
,A->NITE_DAY,A->TEMPERATUR,A->DEPTH,A->VISIBILITY,A->AIR_USED,A->TIME;
```

```
_plineno=0          && set lines to zero
*-- NOEJECT parameter
IF gl_noeject
   IF _peject="BEFORE"
      _peject="NONE"
   ENDIF
   IF _peject="BOTH"
      _peject="AFTER"
   ENDIF
ENDIF

*-- Set-up environment
ON ESCAPE DO Prnabort
IF SET("TALK")="ON"
   SET TALK OFF
   gc_talk="ON"
ELSE
   gc_talk="OFF"
ENDIF
gc_space=SET("SPACE")
SET SPACE OFF
gc_time=TIME()      && system time for predefined field
gd_date=DATE()      && system date "    "    "    "
gl_fandl=.F.        && first and last page flag
gl_prntflg=.T.      && Continue printing flag
gl_widow=.T.        && flag for checking widow bands
gn_length=LEN(gc_heading)  && store length of the HEADING
gn_level=2          && current band being processed
gn_page=_pageno     && grab current page number
gn_pspace=_pspacing && get current print spacing


*-- Set up procedure for page break
gn_atline=_plength - (_pspacing + 1)
ON PAGE AT LINE gn_atline EJECT PAGE

*-- Print Report

PRINTJOB

*-- Initialize summary variables.
r_msum1=0

IF gl_plain
   ON PAGE AT LINE gn_atline DO Pgplain
ELSE
   ON PAGE AT LINE gn_atline DO Pgfoot
ENDIF

DO Pghead

gl_fandl=.T.        && first physical page started
```

```
,A->DIVE_RMKS,B->FNAME,B->MI,B->LNAME,C->TYPE,C->NAME
SET FIELDS ON
SET SAFETY OFF
ERASE TEMP.DBF
COPY TO TEMP.DBF
SET SAFETY ON
*************
ENDIF
***************
return




* Program............: B:\LOG_REPO.FRG
* Date...............: 2-04-91
* Versions...........: dBASE IV, Report 1.1
*
* Notes:
* ------
* Prior to running this procedure with the DO command
* it is necessary use LOCATE because the CONTINUE
* statement is in the main loop.
*
*-- Parameters
PARAMETERS gl_noeject, gl_plain, gl_summary, gc_heading, gc_extra
** The first three parameters are of type Logical.
** The fourth parameter is a string.  The fifth is extra.
PRIVATE _peject, _wrap

*-- Test for no records found
IF EOF() .OR. .NOT. FOUND()
   RETURN
ENDIF

*-- turn word wrap mode off
_wrap=.F.

IF _plength < (_pspacing * 6 + 1) + (_pspacing + 1) + 2
   SET DEVICE TO SCREEN
   DEFINE WINDOW gw_report FROM 7,17 TO 11,62 DOUBLE
   ACTIVATE WINDOW gw_report
   @ 0,1 SAY "Increase the page length for this report."
   @ 2,1 SAY "Press any key ..."
   x=INKEY(0)
   DEACTIVATE WINDOW gw_report
   RELEASE WINDOW gw_report
   RETURN
ENDIF
```

```
DO Rintro

*-- File Loop
DO WHILE FOUND() .AND. .NOT. EOF() .AND. gl_prntflg
   gn_level=0
   *-- Detail lines
   IF gl_summary
      DO Upd_Vars
   ELSE
      DO __Detail
   ENDIF
   gl_widow=.T.         && enable widow checking
   CONTINUE
ENDDO

IF gl_prntflg
   DO Rsumm
   IF _plineno <= gn_atline
      EJECT PAGE
   ENDIF
ELSE
   DO Rsumm
   DO Reset
   RETURN
ENDIF

ON PAGE

ENDPRINTJOB

DO Reset
RETURN
* EOP: B:\LOG_REPO.FRG

*-- Update summary fields and/or calculated fields.
PROCEDURE Upd_Vars
*-- Sum
r_msum1=r_msum1+TIME
RETURN
* EOP: Upd_Vars

*-- Set flag to get out of DO WHILE loop when escape is pressed.
PROCEDURE Prnabort
gl_prntflg=.F.
RETURN
* EOP: Prnabort

PROCEDURE Pghead
?
?? IIF(gl_plain,'' , "Page No." ) AT 0,;
 IIF(gl_plain,'',_pageno) PICTURE "999" AT 9,;
 "Dive log for" AT 22,;
 Fname FUNCTION "T" AT 35,;
 Mi FUNCTION "T" AT 48,;
 Lname FUNCTION "T" AT 51
?

*-- Print HEADING parameter ie. REPORT FORM <name> HEADING <expC>
IF .NOT. gl_plain .AND. gn_length > 0
   ?? gc_heading FUNCTION "I;V"+LTRIM(STR(_rmargin-_lmargin))
   ?
ENDIF
IF .NOT. gl_plain
   ?? gd_date AT 0
   ?
ENDIF
?
?? "Date" AT 0,;
 "Site name or location" AT 9,;
 "Air" AT 40,;
 "Water" AT 49,;
 "Max" AT 60,;
 "Visibility" AT 66,;
 "Time" AT 77
?
?? "Serial" AT 0,;
 "used" AT 40,;
 "temp" AT 49,;
 "depth" AT 60
?
RETURN
* EOP: Pghead

PROCEDURE Rintro
?
RETURN
* EOP: Rintro

PROCEDURE __Detail
IF 3 * gn_pspace < gn_atline - (_pspacing * 6 + 1)
   IF gl_widow .AND. _plineno+3 * gn_pspace > gn_atline + 1
      EJECT PAGE
   ENDIF
ENDIF
DO Upd_Vars
?? Date AT 0,;
 Name FUNCTION "TV30" AT 9,;
 Air_used PICTURE "99999999" AT 40,;
 Temperatur PICTURE "9999999999" AT 49,;
 Depth PICTURE "99999" AT 60,;
 Visibility PICTURE "9999999999" AT 66,;
 Time PICTURE "999.9" AT 77
?
?? Fresh_salt FUNCTION "T" PICTURE "XXXXXXXXXX" AT 9,;
```

```
 "Water," AT 20,;
 Nite_day FUNCTION "T" PICTURE "XXXXXXXX" AT 27,;
 "Dive" AT 36
?
?? Serial PICTURE "999999" AT 0,;
 Dive_rmks FUNCTION "TV30" AT 9
?
RETURN
* EOP: __Detail

PROCEDURE Rsumm
?? "Total time:" AT 62,;
 r_msuml PICTURE "999.9" AT 77
gl_fandl=.F.        && last page finished
?
RETURN
* EOP: Rsumm

PROCEDURE Pgfoot
PRIVATE _box, _pspacing
gl_widow=.F.         && disable widow checking
_pspacing=1
?
IF .NOT. gl_plain
ENDIF
EJECT PAGE
*-- is the page number greater than the ending page
IF _pageno > _pepage
   GOTO BOTTOM
   SKIP
   gn_level=0
ENDIF
IF .NOT. gl_plain .AND. gl_fandl
   _pspacing=gn_pspace
   DO Pghead
ENDIF
RETURN
* EOP: Pgfoot

*-- Process page break when PLAIN option is used.
PROCEDURE Pgplain
PRIVATE _box
EJECT PAGE
RETURN
* EOP: Pgplain

*-- Reset dBASE environment prior to calling report
PROCEDURE Reset
SET SPACE &gc_space.
SET TALK &gc_talk.
ON ESCAPE
ON PAGE
```

```
RETURN
* EOP: Reset




* dBASE IV .QBE file
* QUALLIST.PRG (joins diver.dbf with qual.dbf and stores to temp5.dbf
* for one diver. fields are set for a qualification listing.)
************
PARAMETER BATON
ON ERROR CANCEL
IF .NOT. BATON='GOOD'
  DO VIOLATIO
ELSE
*************
ACCEPT "Enter the diver number of the diver whos qual list you wish to find: ";
to number

SET FIELDS TO
SELECT 1
USE DIVER.DBF AGAIN NOUPDATE
USE QUAL.DBF AGAIN NOUPDATE IN 2 ORDER DIVER_NUM
SET EXACT ON
SET FILTER TO ((A->DIVER_NUM=number)) .AND. FOUND(2)
SET RELATION TO A->DIVER_NUM INTO B
SET SKIP TO B
GO TOP
SET FIELDS TO A->FNAME,A->MI,A->LNAME,B->QUAL_NAME,B->COMPANY,B-
>DATE,B;
->INSTRUCTOR,A->DIVER_NUM
SET FIELDS ON
SET SAFETY OFF
ERASE TEMP5.DBF
COPY TO TEMP5.DBF
SET SAFETY ON
*************
ENDIF
************
return




* Program............: B:\QUALRPT.FRG
* Date...............: 8-04-91
* Versions...........: dBASE IV, Report 1.1
*
* Notes:
* ------
* Prior to running this procedure with the DO command
* it is necessary use LOCATE because the CONTINUE
* statement is in the main loop.
*
*-- Parameters
PARAMETERS gl_noeject, gl_plain, gl_summary, gc_heading, gc_extra
** The first three parameters are of type Logical.
** The fourth parameter is a string.  The fifth is extra.
PRIVATE _peject, _wrap

*-- Test for no records found
IF EOF() .OR. .NOT. FOUND()
   RETURN
ENDIF

*-- turn word wrap mode off
_wrap=.F.

IF _plength < (_pspacing * 6 + 1) + (_pspacing + 1) + 2
   SET DEVICE TO SCREEN
   DEFINE WINDOW gw_report FROM 7,17 TO 11,62 DOUBLE
   ACTIVATE WINDOW gw_report
   @ 0,1 SAY "Increase the page length for this report."
   @ 2,1 SAY "Press any key ..."
   x=INKEY(0)
   DEACTIVATE WINDOW gw_report
   RELEASE WINDOW gw_report
   RETURN
ENDIF

_plineno=0          && set lines to zero
*-- NOEJECT parameter
IF gl_noeject
   IF _peject="BEFORE"
      _peject="NONE"
   ENDIF
   IF _peject="BOTH"
      _peject="AFTER"
   ENDIF
ENDIF

*-- Set-up environment
ON ESCAPE DO Prnabort
IF SET("TALK")="ON"
   SET TALK OFF
   gc_talk="ON"
ELSE
   gc_talk="OFF"
ENDIF
gc_space=SET("SPACE")
SET SPACE OFF
gc_time=TIME()      && system time for predefined field
```

```
gd_date=DATE()      && system date  "   "   "    "
gl_fandl=.F.        && first and last page flag
gl_prntflg=.T.      && Continue printing flag
gl_widow=.T.        && flag for checking widow bands
gn_length=LEN(gc_heading)  && store length of the HEADING
gn_level=2          && current band being processed
gn_page=_pageno     && grab current page number
gn_pspace=_pspacing && get current print spacing


*-- Set up procedure for page break
gn_atline=_plength - (_pspacing + 1)
ON PAGE AT LINE gn_atline EJECT PAGE

*-- Print Report

PRINTJOB

IF gl_plain
   ON PAGE AT LINE gn_atline DO Pgplain
ELSE
   ON PAGE AT LINE gn_atline DO Pgfoot
ENDIF

DO Pghead

gl_fandl=.T.        && first physical page started

*-- File Loop
DO WHILE FOUND() .AND. .NOT. EOF() .AND. gl_prntflg
   gn_level=0
   *-- Detail lines
   IF gl_summary
      DO Upd_Vars
   ELSE
      DO __Detail
   ENDIF
   gl_widow=.T.         && enable widow checking
   CONTINUE
ENDDO

IF gl_prntflg
   DO Rsumm
   IF _plineno <= gn_atline
      EJECT PAGE
   ENDIF
ELSE
   DO Rsumm
   DO Reset
   RETURN
ENDIF

ON PAGE
```

```
ENDPRINTJOB

DO Reset
RETURN
* EOP: B:\QUALRPT.FRG

*-- Update summary fields and/or calculated fields.
PROCEDURE Upd_Vars
RETURN
* EOP: Upd_Vars

*-- Set flag to get out of DO WHILE loop when escape is pressed.
PROCEDURE Prnabort
gl_prntflg=.F.
RETURN
* EOP: Prnabort

PROCEDURE Pghead
?
?? IIF(gl_plain,'' , "Page No." ) AT 0,;
 IIF(gl_plain,'',_pageno) PICTURE "999" AT 9,;
 "Qual list for:" AT 21
?

*-- Print HEADING parameter ie. REPORT FORM <name> HEADING <expC>
IF .NOT. gl_plain .AND. gn_length > 0
   ?? gc_heading FUNCTION "I;V"+LTRIM(STR(_rmargin-_lmargin))
   ?
ENDIF
?? IIF(gl_plain,'',gd_date) AT 0,;
 Fname FUNCTION "T" AT 18,;
 Mi FUNCTION "T" AT 31,;
 Lname FUNCTION "T" AT 34
?
?
?
?? "QUAL_NAME" AT 5,;
 "COMPANY" AT 17,;
 "DATE" AT 29,;
 "INSTRUCTOR" AT 39
?
RETURN
* EOP: Pghead


PROCEDURE __Detail
IF gn_pspace < gn_atline - (_pspacing * 6 + 1)
   IF gl_widow .AND. _plineno+gn_pspace > gn_atline + 1
      EJECT PAGE
   ENDIF
ENDIF
DO Upd_Vars
```

```
?? Qual_name FUNCTION "T" AT 5,;
 Company FUNCTION "T" AT 17,;
 Date AT 29,;
 Instructor FUNCTION "T" AT 39
?
RETURN
* EOP: __Detail

PROCEDURE Rsumm
gl_fandl=.F.        && last page finished
?
RETURN
* EOP: Rsumm

PROCEDURE Pgfoot
PRIVATE _box, _pspacing
gl_widow=.F.        && disable widow checking
_pspacing=1
?
IF .NOT. gl_plain
ENDIF
EJECT PAGE
*-- is the page number greater than the ending page
IF _pageno > _pepage
   GOTO BOTTOM
   SKIP
   gn_level=0
ENDIF
IF .NOT. gl_plain .AND. gl_fandl
   _pspacing=gn_pspace
   DO Pghead
ENDIF
RETURN
* EOP: Pgfoot

*-- Process page break when PLAIN option is used.
PROCEDURE Pgplain
PRIVATE _box
EJECT PAGE
RETURN
* EOP: Pgplain

*-- Reset dBASE environment prior to calling report
PROCEDURE Reset
SET SPACE &gc_space.
SET TALK &gc_talk.
ON ESCAPE
ON PAGE
RETURN
* EOP: Reset
```

LIST OF REFERENCES

Murray, W., "Security Considerations for Personal Computers," *IBM Systems Journal*, pg.27, v.23, no.3, 1984.

Stephenson, P., "Personal and Private," *Byte*, pg.286, June 1989.

Brown, B., "The Small Data Center," *Byte*, pg.286, June 1989.

Pfleeger, C., *Security in Computing*, Prentice-Hall, Inc., 1989.

Giladi, R. and Zviran, M., *Centralizing the Data, Distributing the Processing*, Working Paper No. 89-02, Naval Postgraduate School, January 1989.

National Computer Security Center, *Personal Computer Security Considerations*, NCSC Pub WA-002-85, 1985.

National Telecommunications and Information Systems Security Commitee, *Office Automation Security Guide*, NTISS COMPUSEC / 1-87, 1987.

Post, G. and Kievit, K., "Accessibility vs. Security: A Look at the Demand for Computer Security," *Computers and Security*, Vol. 10, No. 4, June 1991,Elsevier Science Publishers B. V.

Gogan, J., "Should PCs Be Personally Allocated?", *Journal of Management Information Systems*, Spring 1991, Vol. 7, No. 4, 1991.

Tanenbaum, A., *Operating Systems: Design and Implementation*, pp.4-5, Prentice-Hall, Inc., 1987.

Tanenbaum, A., *Structured Computer Organization*, Prentice-Hall, Inc., 1990.

Zarger, C., "Is Your PC Secure?" *Mechanical Engineering*, pg. 57, March 1988.

Mensching, J. and Adams, D., *Managing an Information System*, Prentice Hall, Inc., 1991.

BIBLIOGRAPHY

Awad, E., *Management Information Systems*, Benjamin Cummings, Inc., 1988.

Bakst, S., "Beware of Potholes on the Path to PC Security," *The Office*, June 1990.

Boebert, W., Kain, R. and Young, W., "Secure Computing: The Secure Ada Target Approach," *Scientific Honeyweller*, July 1985.

Brown, B., "The Small Data Center," *Byte*, pg.286, June 1989.

Chorley, B. and Price, W., "An Intelligent Token For Secure Transactions," *Security and Protection in Information Systems*, Elsevier Science Publishers B. V., 1989.

U. S. Dept. of Defense, *Trusted Computing Sys. Evaluation Criteria*, DOD 5200.28STD, Dec 85.

Giladi, R. and Zviran, M., *Centralizing the Data, Distributing the Processing*, Working Paper No. 89-02, Naval Postgraduate School, January 1989.

Gogan, J., "Should PCs Be Personally Allocated?", *Journal of Management Information Systems*, Spring 1991, Vol. 7, No. 4, 1991.

Mehrmann, L. and Amery, C., "Security Practices for Information Systems Networks," *Security and Protection in Information Systems*, Elsevier Science Publishers B. V., 1989.

Mensching, J. and Adams, D., *Managing an Information System*, Prentice Hall, Inc., 1991.

Murray, W., "Security Considerations for Personal Computers," *IBM Systems Journal*, v.23, no.3, 1984.

Murray, W., "Security in Advanced Applications and Environments," *Security and Protection in Information Systems*, Elsevier Science Publishers B. V., 1989.

National Computer Security Center, *Personal Computer Security Considerations*, NCSC Pub WA-002-85, 1985.

Pfleeger, C., *Security in Computing*, Prentice-Hall, Inc., 1989.

Post, G. and Kievit, K. "Accessibility vs. Security: A Look at the Demand for Computer Security," *Computers and Security*, Vol. 10, No. 4, June 1991, Elsevier Science Publishers B. V.

Schultz, J., "Low Cost Security for Personal Computers," *Signal*, November 1989.

Stephenson, P., "Personal and Private," *Byte*, June 1989.

Summers, R., "An Overview of Computer Security," *IBM Systems Journal*, v.23, no.4, 1984.

Tanenbaum, A., *Operating Systems:Design and Implementation*, Prentice-Hall, Inc., 1987.

Tanenbaum, A., *Structured Computer Organization*, Prentice-Hall, Inc., 1990.

Walker, S., "Network Security Overview", paper presented at the 1985 Symposium on Security and Privacy, 1985.

Zarger, C., "Is Your PC Secure?" *Mechanical Engineering*, pg. 57, March 1988.

INITIAL DISTRIBUTION LIST

1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Library, Code 52 Naval Postgraduate School Monterey, California 93943-5002	2
3.	Prof. Moshe Zviran, Code AS/ZV Naval Postgraduate School Monterey, California 93943-5000	1
4.	Prof. William J. Haga, Code AS/HG Naval Postgraduate School Monterey, California 93943-5000	1

INITIAL DISTRIBUTION LIST

1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Library, Code 52 Naval Postgraduate School Monterey, California 93943-5002	2
3.	Prof. Moshe Zviran, Code AS/ZV Naval Postgraduate School Monterey, California 93943-5000	1
4.	Prof. William J. Haga, Code AS/HG Naval Postgraduate School Monterey, California 93943-5000	1